ВТОРОЕ ИЗДАНИЕ

НЕЙРОННЫЕ СЕТИ

ПОЛНЫЙ КУРС

САЙМОН ХАЙКИН

NEURAL NETWORKS

A Comprehensive Foundation

Second Edition

Simon Haykin
McMaster University
Hamilton, Ontario, Canada

Prentice Hall
Upper Saddle River, New Jersey 07458

НЕЙРОННЫЕ СЕТИ

Полный курс

Второе издание

Саймон Хайкин
Университет McMaster
Гамильтон, Онтарио, Канада

Москва • Санкт-Петербург • Киев
2006

ББК 32.973.26-018.2.75
Х15
УДК 681.3.07

Издательский дом "Вильямс"
Зав. редакцией *С.Н. Тригуб*

Перевод с английского. д.т.н. *Н.Н. Куссуль*, к.т.н. *А.Ю. Шелестова*
Под редакцией д.т.н. *Н.Н. Куссуль*

По общим вопросам обращайтесь в Издательский дом "Вильямс"
по адресу: info@williamspublishing.com, http://www.williamspublishing.com
115419, Москва, а/я 783; 03150, Киев, а/я 152

Хайкин, Саймон.

Х15 Нейронные сети: полный курс, 2-е издание. : Пер. с англ. — М. : Издательский дом "Вильямс", 2006. — 1104 с. : ил. — Парал. тит. англ.

ISBN 5-8459-0890-6 (рус.)

В книге рассматриваются основные парадигмы искусственных нейронных сетей. Представленный материал содержит строгое математическое обоснование всех нейросетевых парадигм, иллюстрируется примерами, описанием компьютерных экспериментов, содержит множество практических задач, а также обширную библиографию.

В книге также анализируется роль нейронных сетей при решении задач распознавания образов, управления и обработки сигналов. Структура книги очень удобна для разработки курсов обучения нейронным сетям и интеллектуальным вычислениям.

Книга будет полезна для инженеров, специалистов в области компьютерных наук, физиков и специалистов в других областях, а также для всех тех, кто интересуется искусственными нейронными сетями.

ББК 32.973.26-018.2.75

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства Prentice Hall, Inc.

Authorized translation from the English language edition published by Prentice Hall, Copyright © 1998

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Russian language edition was published by Williams Publishing House according to the Agreement with R&I Enterprises International, Copyright © 2006

ISBN 5-8459-0890-6 (рус.)
ISBN 0-13-273350-1 (англ.)

© Издательский дом "Вильямс", 2006
© Prentice Hall, Inc., 1999

Оглавление

Предисловие 22

1. Введение 31

2. Процессы обучения 89

3. Однослойный персептрон 172

4. Многослойный персептрон 219

5. Сети на основе радиальных базисных функций 341

6. Машины опорных векторов 417

7. Ассоциативные машины 458

8. Анализ главных компонентов 509

9. Карты самоорганизации 573

10. Модели на основе теории информации 622

11. Стохастические машины и их аппроксимации в статистической механике 691

12. Нейродинамическое программирование 760

13. Временная обработка с использованием сетей прямого распространения 799

14. Нейродинамика 835

15. Динамически управляемые рекуррентные сети 919

16. Заключение 989

Библиография 996

Предметный указатель 1069

Содержание

Предисловие 22

Благодарности 25
Важные символы 27
Открытые и закрытые интервалы 29
Минимумы и максимумы 29

1. Введение 31

1.1. Что такое нейронные сети 31
Преимущества нейронных сетей 33
1.2. Человеческий мозг 37
1.3. Модели нейронов 42
Типы функций активации 45
Стохастическая модель нейрона 48
1.4. Представление нейронных сетей с помощью направленных графов 49
1.5. Обратная связь 52
1.6. Архитектура сетей 55
Однослойные сети прямого распространения 56
Многослойные сети прямого распространения 56
Рекуррентные сети 57
1.7. Представление знаний 58
Как встроить априорную информацию в структуру нейронной сети 64
Как встроить инварианты в структуру нейронной сети 65
1.8. Искусственный интеллект и нейронные сети 71
1.9. Историческая справка 75
Задачи 84
Модели нейрона 84
Сетевые архитектуры 86
Представление знаний 88

2. Процессы обучения 89

2.1. Введение 89
Структура главы 90
2.2. Обучение, основанное на коррекции ошибок 91
2.3. Обучение на основе памяти 93
2.4. Обучение Хебба 95

Усиление и ослабление синаптической связи 97
Математические модели предложенного Хеббом механизма модификации синаптической связи 98
2.5. Конкурентное обучение 101
2.6. Обучение Больцмана 104
2.7. Задача присваивания коэффициентов доверия 106
2.8. Обучение с учителем 107
2.9. Обучение без учителя 108
Обучение с подкреплением, или нейродинамическое программирование 109
Обучение без учителя 110
2.10. Задачи обучения 111
Ассоциативная память 111
Распознавание образов 113
Аппроксимация функций 114
Управление 116
Фильтрация 118
Формирование диаграммы направленности 120
2.11. Память 122
Память в виде матрицы корреляции 127
Извлечение из памяти 129
2.12. Адаптация 132
2.13. Статистическая природа процесса обучения 134
Дилемма смещения и дисперсии 138
2.14. Теория статистического обучения 140
Некоторые основные определения 142
Принцип минимизации эмпирического риска 143
VC-измерение 146
Важность VC-измерения и его оценка 149
Конструктивные, независимые от распределения пределы обобщающей способности обучаемых машин 151
Минимизация структурного риска 154
2.15. Вероятностно-корректная в смысле аппроксимации модель обучения 156
Сложность обучающего множества 159
Вычислительная сложность 160
2.16. Резюме и обсуждение 161
Задачи 163
Правила обучения 163
Парадигмы обучения 166
Память 167
Адаптация 168
Статистическая теория обучения 169

3. Однослойный персептрон 171

3.1. Введение 171
Структура главы 172
3.2. Задача адаптивной фильтрации 173
3.3. Методы безусловной оптимизации 175
Метод наискорейшего спуска 177
Метод Ньютона 179
Метод Гаусса–Ньютона 181
3.4. Линейный фильтр, построенный по методу наименьших квадратов 183
Фильтр Винера как ограниченная форма линейного фильтра, построенного по методу наименьших квадратов, для эргодической среды 184
3.5. Алгоритм минимизации среднеквадратической ошибки 185
Граф передачи сигнала для алгоритма минимизации среднеквадратической ошибки 187
Условия сходимости алгоритма LMS 188
Преимущества и недостатки алгоритма LMS 190
3.6. Графики процесса обучения 191
3.7. Изменение параметра скорости обучения по модели отжига 193
3.8. Персептрон 194
3.9. Теорема о сходимости персептрона 196
3.10. Взаимосвязь персептрона и байесовского классификатора в гауссовой среде 204
Байесовский классификатор 204
Байесовский классификатор и распределение Гаусса 207
3.11. Резюме и обсуждение 210
Задачи 212
Безусловная оптимизация 212
Алгоритм LMS 213
Персептрон Розенблатта 216

4. Многослойный персептрон 219

4.1. Введение 219
Структура главы 221
4.2. Вводные замечания 222
Обозначения 224
4.3. Алгоритм обратного распространения 225
Случай 1. Нейрон j — выходной узел 229
Случай 2. Нейрон j — скрытый узел 229
Два прохода вычислений 232
Функция активации 233

Скорость обучения 235
Последовательный и пакетный режимы обучения 238
Критерий останова 240
4.4. Алгоритм обратного распространения в краткой форме 241
4.5. Задача XOR 243
4.6. Эвристические рекомендации по улучшению работы алгоритма обратного распространения 245
4.7. Представление выхода и решающее правило 253
4.8. Компьютерный эксперимент 256
Байесовская граница решений 257
Экспериментальное построение оптимального многослойного персептрона 260
4.9. Извлечение признаков 268
Связь с линейным дискриминантом Фишера 272
4.10. Обратное распространение ошибки и дифференцирование 274
Матрица якобиана 275
4.11. Гессиан 276
4.12. Обобщение 278
Достаточный объем примеров обучения для корректного обобщения 279
4.13. Аппроксимация функций 281
Теорема об универсальной аппроксимации 282
Пределы ошибок аппроксимации 283
"Проклятие размерности" 285
Практические соображения 286
4.14. Перекрестная проверка 288
Выбор модели 289
Метод обучения с ранним остановом 291
Варианты метода перекрестной проверки 294
4.15. Методы упрощения структуры сети 295
Регуляризация сложности 296
Упрощение структуры сети на основе Гессиана 299
4.16. Преимущества и ограничения обучения методом обратного распространения 304
Связность 306
Извлечение признаков 307
Аппроксимация функций 309
Вычислительная эффективность 310
Анализ чувствительности 310
Робастность 311
Сходимость 311
Локальные минимумы 312
Масштабирование 313
4.17. Ускорение сходимости процесса обучения методом обратного распространения 315

4.18. Обучение с учителем как задача оптимизации 316
Метод сопряженных градиентов 319
Нелинейный алгоритм сопряженных градиентов в сжатом виде 326
Квазиньютоновкие методы 327
Сравнение квазиньютоновских методов с методом сопряженных градиентов 329
4.19. Сети свертки 330
4.20. Резюме и обсуждение 333
Задачи 335
Задачи XOR 335
Обучение методом обратного распространения 335
Перекрестная проверка 336
Приемы упрощения сети 337
Ускорение сходимости алгоритма обратного распространения 337
Методы оптимизации второго порядка 338
Компьютерное моделирование 338

5. Сети на основе радиальных базисных функций 341

5.1. Введение 341
Структура главы 342
5.2. Теорема Ковера о разделимости множеств 343
Разделяющая способность поверхности 347
5.3. Задача интерполяции 349
Теорема Мичелли 352
5.4. Обучение с учителем как плохо обусловленная задача восстановления гиперповерхности 353
5.5. Теория регуляризации 355
Дифференциал Фреше функционала Тихонова 358
Уравнение Эйлера–Лагранжа 360
Функция Грина 361
Решение задачи регуляризации 363
Определение коэффициентов разложения 364
Многомерные функции Гаусса 367
5.6. Сети регуляризации 369
5.7. Обобщенные сети на основе радиальных базисных функций 371
Взвешенная норма 373
Рецептивные поля 375
5.8. Задача XOR (повторное рассмотрение) 376
5.9. Оценивание параметра регуляризации 378
Среднеквадратическая ошибка 379
Обобщенная перекрестная проверка 382
Оптимальное свойство обобщенной функции перекрестной проверки $V\lambda$ 384
Заключительные комментарии 385
5.10. Свойства аппроксимации сетей RBF 385

Универсальная теорема об аппроксимации 386
"Проклятие размерности" (продолжение) 386
Связь между сложностью обучающего множества, вычислительной сложностью и эффективностью обобщения 388
5.11. Сравнение сетей RBF и многослойных персептронов 389
5.12. Регрессия ядра и ее связь с сетями RBF 390
Многомерное распределение Гаусса 395
5.13. Стратегии обучения 396
Случайный выбор фиксированных центров 396
Выбор центров на основе самоорганизации 399
Выбор центров с учителем 401
Строгая интерполяция с регуляризацией 403
5.14. Компьютерное моделирование: классификация образов 405
5.15. Резюме и обсуждение 408
Задачи 409
Радиальные базисные функции 409
Сети регуляризации 410
Порядок аппроксимации 413
Оценка ядра 414
Выбор центров с учителем 414
Компьютерное моделирование 415

6. Машины опорных векторов 417

6.1. Введение 417
Структура главы 418
6.2. Оптимальная гиперплоскость для линейно-разделимых образов 419
Квадратичная оптимизация и поиск оптимальной гиперплоскости 422
Статистические свойства оптимальной гиперплоскости 425
6.3. Оптимальная гиперплоскость для неразделимых образов 426
6.4. Как создать машину опорных векторов для задачи распознавания образов 431
Ядро скалярного произведения 433
Теорема Мерсера 434
Оптимальная архитектура машины опорных векторов 436
Примеры машин опорных векторов 437
6.5. Пример: задача XOR (продолжение) 438
6.6. Компьютерное моделирование 442
Заключительные замечания 443
6.7. ε-нечувствительные функции потерь 444
6.8. Машины опорных векторов для задач нелинейной регрессии 445
6.9. Резюме и обсуждение 449
Задачи 453
Оптимальная разделяющая гиперплоскость 453
Ядро скалярного произведения 454

Классификация множеств 455
Нелинейная регрессия 455
Преимущества и недостатки 456
Компьютерное моделирование 457

7. Ассоциативные машины 458

7.1. Введение 458
Структура главы 459
7.2. Усреднение по ансамблю 460
7.3. Компьютерный эксперимент 1 464
7.4. Метод усиления 465
Усиление за счет фильтрации 466
Алгоритм адаптивного усиления AdaBoost 470
Изменение ошибки 473
7.5. Компьютерный эксперимент 2 474
7.6. Ассоциативная гауссова модель смешения 476
Вероятностная порождающая модель 478
Модель смешения мнений экспертов 479
7.7. Модель иерархического смешения мнений экспертов 484
7.8. Выбор модели с использованием стандартного дерева решений 486
Алгоритм CART 487
Использование алгоритма CART для инициализации модели HMA 489
7.9. Априорные и апостериорные вероятности 490
7.10. Оценка максимального подобия 492
7.11. Стратегии обучения для модели HME 495
7.12. Алгоритм EM 497
7.13. Применение алгоритма EM к модели HME 498
7.14. Резюме и обсуждение 503
Задачи 505
Усреднение по ансамблю 505
Усиление 505
Смешение мнений экспертов 505
Иерархическое смешение мнений экспертов 506
Алгоритм EM и его применение в модели HME 506

8. Анализ главных компонентов 509

8.1. Введение 509
Структура главы 510
8.2. Некоторые интуитивные принципы самоорганизации 510
Анализ признаков на основе самоорганизации 513
8.3. Анализ главных компонентов 514
Структура анализа главных компонентов 516
Основные представления данных 520

Сокращение размерности 520
8.4. Фильтр Хебба для выделения максимальных собственных значений 523
Матричная формулировка алгоритма 527
Теорема об асимптотической устойчивости 528
Анализ устойчивости фильтра для извлечения максимального собственного значения 530
Общие свойства фильтра Хебба для извлечения максимального собственного значения 535
8.5. Анализ главных компонентов на основе фильтра Хебба 537
Исследование сходимости 541
Оптимальность обобщенного алгоритма Хебба 542
Алгоритм GHA в сжатом виде 543
8.6. Компьютерное моделирование: кодирование изображений 544
8.7. Адаптивный анализ главных компонентов с использованием латерального торможения 546
Интенсивность обучения 556
Алгоритм APEX в сжатом виде 557
8.8. Два класса алгоритмов PCA 558
Подпространство главных компонентов 558
8.9. Пакетный и адаптивный методы вычислений 559
8.10. Анализ главных компонентов на основе ядра 561
Алгоритм PCA на основе ядра в сжатом виде 565
8.11. Резюме и обсуждение 567
Задачи 570
Фильтр Хебба для извлечения максимального собственного значения 570
Анализ главных компонентов на основе правила Хебба 571
PCA на основе ядра 572

9. Карты самоорганизации 573

9.1. Введение 573
Структура главы 574
9.2. Две основные модели отображения признаков 575
9.3. Карты самоорганизации 577
Процесс конкуренции 579
Процесс кооперации 580
Процесс адаптации 583
Два этапа адаптивного процесса: упорядочивание и сходимость 585
9.4. Краткое описание алгоритма SOM 586
9.5. Свойства карты признаков 588
9.6. Компьютерное моделирование 597
Двумерная решетка, полученная на основе двумерного распределения 597
Одномерная решетка на основе двумерного распределения 599
Описание параметров моделирования 600

9.7. Квантование вектора обучения 602
9.8. Компьютерное моделирование: адаптивная классификация множеств 604
9.9. Иерархическая квантизация векторов 606
9.10. Контекстные карты 611
9.11. Резюме и обсуждение 613
Задачи 615
Алгоритм SOM 615
Квантизация векторов обучения 616
Компьютерные эксперименты 617

10. Модели на основе теории информации 622

10.1. Введение 622
Структура главы 623
10.2. Энтропия 623
Дифференциальная энтропия непрерывной случайной переменной 627
Свойства дифференциальной энтропии 628
10.3. Принцип максимума энтропии 629
10.4. Взаимная информация 632
Взаимная информация непрерывных случайных переменных 635
10.5. Дивергенция Кулбека–Лейблера 636
Декомпозиция Пифагора 638
10.6. Взаимная информация как оптимизируемая целевая функция 640
10.7. Принцип максимума взаимной информации 641
10.8. Принцип Infomax и уменьшение избыточности 646
Моделирование систем восприятия 646
10.9. Пространственно связные признаки 649
10.10. Пространственно несвязные признаки 652
10.11. Анализ независимых компонентов 654
Критерий статистической независимости 659
Определение дифференциальной энтропии $h(Y)$ 660
Определение граничной энтропии $\tilde{h}(Y_i)$ 660
Функция активации 664
Алгоритм обучения для ICA 666
Свойство эквивариантности 668
Условия устойчивости 670
Условия сходимости 672
10.12. Компьютерное моделирование 672
10.13. Оценка максимального правдоподобия 675
Связь между максимальным подобием и анализом независимых компонентов 677
10.14. Метод максимальной энтропии 678
Алгоритм обучения для слепого разделения источников 682
10.15. Резюме и обсуждение 684

Задачи 686
Принцип максимума энтропии 686
Взаимная информация 686
Принцип Infomax 687
Анализ независимых компонентов 688
Метод максимальной энтропии 690

11. Стохастические машины и их аппроксимации в статистической механике 691

11.1. Введение 691
Структура главы 692
11.2. Статистическая механика 692
Свободная энергия и энтропия 694
11.3. Цепи Маркова 695
Вероятности перехода 696
Свойства рекуррентности 698
Несократимые цепи Маркова 698
Эргодические цепи Маркова 699
Сходимость к стационарным распределениям 700
Классификация состояний 703
Принцип детального баланса 703
11.4. Алгоритм Метрополиса 704
Выбор вероятности перехода 705
11.5. Метод моделирования отжига 707
Расписание отжига 709
Моделирование отжига для комбинаторной оптимизации 710
11.6. Распределение Гиббса 711
11.7. Машина Больцмана 713
Квантование Гиббса и моделирование отжига в машине Больцмана 715
Правило обучения Больцмана 718
Потребность в отрицательной фазе и ее применение 721
11.8. Сигмоидальные сети доверия 722
Фундаментальные свойства сигмоидальных сетей доверия 722
Обучение в сигмоидальных сетях доверия 724
11.9. Машина Гельмгольца 728
11.10. Теория среднего поля 730
11.11. Детерминированная машина Больцмана 733
11.12. Детерминированные сигмоидальные сети доверия 734
Нижняя граница функции логарифмического правдоподобия 735
Процедура обучения для аппроксимации среднего поля сигмоидальной сети доверия 738
11.13. Детерминированный отжиг 742
Кластеризация посредством детерминированного отжига 743
Аналогия с алгоритмом EM 748
11.14. Резюме и обсуждение 748

Задачи 752
Цепи Маркова 752
Приемы моделирования 752
Машина Больцмана 754
Сигмоидальные сети доверия 757
Машина Гельмгольца 757
Детерминированная машина Больцмана 758
Детерминированная сигмоидальная сеть доверия 758
Детерминированный отжиг 758

12. Нейродинамическое программирование 760

12.1. Введение 760
Структура главы 762
12.2. Марковский процесс принятия решений 762
Постановка задачи 765
12.3. Критерий оптимальности Беллмана 766
Алгоритм динамического программирования 767
Уравнение оптимальности Беллмана 768
12.4. Итерация по стратегиям 770
12.5. Итерация по значениям 773
12.6. Нейродинамическое программирование 778
12.7. Приближенный алгоритм итерации по стратегиям 780
12.8. Q-обучение 784
Теорема о сходимости 786
Приближенное Q-обучение 787
Исследование 788
12.9. Компьютерный эксперимент 790
12.10. Резюме и обсуждение 793
Задачи 796
Критерий оптимальности Беллмана 796
Итерация по стратегиям 798
Итерация по значениям 798
Q-обучение 798

13. Временная обработка с использованием сетей прямого распространения 799

13.1. Введение 799
Структура главы 800
13.2. Структуры кратковременной памяти 801
Память на основе линии задержки с отводами 803
Гамма-память 804
13.3. Сетевые архитектуры для временной обработки 806
NETtalk 806
Нейронные сети с задержкой по времени 807

13.4. Фокусированные сети прямого распространения с задержкой по времени 809
13.5. Компьютерное моделирование 812
13.6. Универсальная теорема миопического отображения 813
13.7. Пространственно-временные модели нейрона 815
Аддитивная модель 818
13.8. Распределенные сети прямого распространения с задержкой по времени 820
13.9. Алгоритм обратного распространения во времени 821
Ограничения причинности 827
13.10. Резюме и обсуждение 829
Задачи 830
Фокусированные TLFN 830
Пространственно-временные модели нейронов 831
Обратное распространение во времени 831
Компьютерное моделирование 832

14. Нейродинамика 835

14.1. Введение 835
Структура главы 836
14.2. Динамические системы 837
Пространство состояний 838
Условие Лившица 840
Теорема о дивергенции 841
14.3. Устойчивость состояний равновесия 842
Определения устойчивости 843
Теоремы Ляпунова 846
14.4. Аттракторы 848
Гиперболические аттракторы 849
14.5. Нейродинамические модели 849
Аддитивная модель 850
Связанная модель 853
14.6. Управление аттракторами как парадигма рекуррентных сетей 854
14.7. Модель Хопфилда 856
Соотношение между устойчивыми состояниями дискретной и непрерывной версии модели Хопфилда 860
Дискретная модель Хопфилда как ассоциативная память 862
Ложные состояния 870
Емкость сети Хопфилда 871
14.8. Компьютерное моделирование 1 876
14.9. Теорема Коэна–Гроссберга 880
Модель Хопфилда как частный случай теоремы Коэна–Гроссберга 883
14.10. Модель BSB 884
Функция Ляпунова модели BSB 885

Динамика модели BSB 888
Кластеризация 889
14.11. Компьютерное моделирование 2 891
14.12. Странные аттракторы и хаос 893
Инвариантные характеристики хаотической динамики 894
14.13. Динамическое восстановление 899
Рекурсивное прогнозирование 901
Две возможные формулировки рекурсивного прогнозирования 903
Динамическое восстановление как плохо обусловленная задача фильтрации 903
14.14. Компьютерное моделирование 3 904
Выбор параметров m и λ 907
14.15. Резюме и обсуждение 908
Задачи 912
Динамические системы 912
Модели Хопфилда 912
Теорема Коэна–Гроссберга 917

15. Динамически управляемые рекуррентные сети 919

15.1. Введение 919
Структура главы 920
15.2. Архитектуры рекуррентных сетей 921
Рекуррентная модель "вход-выход" 921
Модель в пространстве состояний 923
Рекуррентный многослойный персептрон 925
Сеть второго порядка 925
15.3. Модель в пространстве состояний 928
Управляемость и наблюдаемость 930
Локальная управляемость 932
Локальная наблюдаемость 934
15.4. Нелинейная автогрессия с внешней моделью входов 936
15.5. Вычислительная мощность рекуррентных сетей 937
15.6. Алгоритмы обучения 941
Некоторые эвристики 942
15.7. Обратное распространение во времени 943
Обратное распространение по эпохам во времени 945
Усеченное обратное распространение во времени 946
Некоторые практические соглашения 947
15.8. Рекуррентное обучение в реальном времени 949
Усиление учителем 955
15.9. Фильтр Калмана 956
Фильтр Калмана на основе квадратного корня 960
15.10. Несвязный расширенный фильтр Калмана 960

Искусственный шум процесса 964
Полное описание алгоритма DEKF 965
Вычислительная сложность 965
15.11. Компьютерное моделирование 965
15.12. Обращение в нуль градиентов в рекуррентных сетях 968
Долгосрочные зависимости 971
15.13. Системная идентификация 973
Идентификация систем с использованием модели в пространстве состояний 974
Модель в терминах "вход-выход" 976
15.14. Адаптивное управление на основе эталонной модели 977
15.15. Резюме и обсуждение 979
Задачи 982
Модель в пространстве состояний 982
Модель нелинейной авторегрессии с экзогенными входами (NARX) 983
Алгоритм BPTT 985
Алгоритм рекуррентного обучения в реальном времени 985
Алгоритм несвязной расширенной фильтрации Калмана 986
Рекуррентные сети второго порядка 987

16. Заключение **989**

16.1. Интеллектуальные системы 990

Библиография **996**

Предметный указатель **1069**

Огромному количеству исследователей нейронных сетей за их творческий вклад,

всем критикам за их замечания,

моим многочисленным аспирантам за их живой интерес

и моей супруге Нэнси за ее терпение и понимание

Предисловие

Нейронные сети, или, точнее, искусственные нейронные сети, представляют собой технологию, уходящую корнями во множество дисциплин: нейрофизиологию, математику, статистику, физику, компьютерные науки и технику. Они находят свое применение в таких разнородных областях, как моделирование, анализ временных рядов, распознавание образов, обработка сигналов и управление благодаря одному важному свойству — способности *обучаться* на основе данных при участии учителя или без его вмешательства.

Эта книга предлагает исчерпывающее описание нейронных сетей, учитывая многодисциплинарную природу этого предмета. Представленный в книге материал насыщен примерами, описанием компьютерных экспериментов, содержит множество задач по каждому разделу, а также обширную библиографию.

Книга состоит из четырех частей, организованных следующим образом.

Вводная часть (главы 1 и 2). В главе 1 на качественном уровне рассматривается сущность нейронных сетей, их свойства, возможности и связь с другими дисциплинами искусственного интеллекта. Глава завершается некоторыми историческими заметками. В главе 2 представлен обзор основных подходов к обучению и их статистических свойств. В этой главе вводится важное понятие *измерения Вапника–Червоненкиса*, используемое в качестве меры возможности обучаемой машины реализовать семейство функций классификации.

Обучение с учителем — основному предмету обсуждения посвящены главы 3–7. В главе 3 рассматривается простейший класс нейронных сетей — сети с одним или несколькими выходными нейронами, не содержащие скрытого слоя. В этой главе рассматриваются алгоритм метода наименьших квадратов (очень популярный при создании линейных адаптивных фильтров) и теорема о сходимости персептрона. В главе 4 приводится исчерпывающее описание *многослойного персептрона*, обучаемого на основе *алгоритма обратного распространения*. Этот алгоритм (представляющий собой обобщение метода наименьших квадратов) стал основной рабочей лошадкой нейронных сетей. В главе 5 содержится подробное математическое представление другого класса многослойных нейронных сетей — *сетей на основе радиальных базисных функций* (RBF-сетей), включающих отдельный слой базисных функций. Основной акцент в этой главе делается на роли теории регуляризации при создании RBF-сетей. В гла-

ве 6 рассматривается сравнительно новый класс обучаемых систем, известных под названием *машины опорных векторов*. Теория таких систем базируется на материале главы 2, посвященном статистической теории обучения. Вторая часть книги завершается главой 7, в которой обсуждаются *ассоциативные машины*, включающие в качестве своих компонентов несколько отдельно обучаемых систем. В этой главе рассматриваются три различных подхода к построению ассоциативных машин — методы *усреднения по ансамблю*, *усиления* и *иерархического смешения мнений экспертов*.

Обучению без учителя посвящены главы 8–12. В главе 8 *Хеббовское правило обучения* применяется для решения задачи *анализа главных компонентов*. В главе 9 описывается еще одна форма обучения на основе самоорганизации, — *конкурентное обучение*, — применяемое для построения вычислительных отображений, получивших название *карт самоорганизации*. Эти две парадигмы отличаются в основном правилами обучения, уходящими корнями в область нейробиологии. Глава 10 посвящена *теории информации*, применяемой для создания алгоритмов обучения без учителя, а также их применению к решению задач *моделирования*, *обработки изображений* и *анализа независимых компонентов*. В главе 11 рассматриваются вопросы обучения машин на основе принципов *статистической механики*. Эта тематика близко примыкает к материалу предыдущей главы, посвященной вопросам теории информации. В главе 12, которая завершает третью часть книги, вводится понятие *динамического программирования* и анализируется его взаимосвязь с *обучением с подкреплением*.

Нелинейные динамические системы — предмет изучения глав 13–15. В главе 13 описывается класс динамических систем на основе краткосрочной памяти и многослойных нейросетевых структур прямого распространения. В главе 14 основное внимание уделяется вопросам устойчивости, возникающим в нелинейных динамических системах при наличии *обратной связи*. Приводятся примеры *ассоциативной памяти*. В главе 15 рассматривается еще один класс нелинейных динамических систем — *рекуррентные сети*, основанные на использовании обратной связи при построении отображения вход-выход.

В заключении кратко анализируется роль нейронных сетей при создании *интеллектуальных машин* распознавания образов, управления и обработки сигналов.

Структура книги обеспечивает большую гибкость при разработке различных курсов обучения нейронным сетям. Выбор конкретных тем для изучения должен определяться только интересами аудитории, использующей эту книгу.

В книге описывается пятнадцать компьютерных экспериментов, тринадцать из которых рассчитаны на использование MATLAB. Файлы для выполнения этих экспериментов в среде MATLAB можно найти по адресу `ftp://ftp.mathworks.com/pub/books/haykin` или `http://www.mathworks.com/books/`

Во втором случае пользователь должен щелкнуть на ссылке Neural/Fuzzy, а затем — на названии книги. Такой подход обеспечивает более удобный интерфейс пользователя.

Каждая глава завершается несколькими упражнениями. Многие из этих задач являются творческими и призваны не только проверить качество усвоения материала, но и подтолкнуть к дальнейшему развитию соответствующей тематики. Решения всех задач содержатся в руководстве, копии которого могут получить преподаватели, использующие в своих курсах настоящую книгу. Для этого они должны обратиться в издательство Prentice Hall.

Книга рассчитана на инженеров, специалистов в области компьютерных наук и физиков. Автор надеется, что книга будет интересна также специалистам в других областях, таких как психология и нейробиология.

Саймон Хайкин,

Гамильтон, Онтарио,

февраль 1998 г.

Благодарности

Автор выражает глубокую благодарность многим рецензентам, которые потратили свое время на чтение этой книги или отдельной ее части. Хочу сказать большое спасибо доктору Кеннету Роузу (Kenneth Rose) из университета штата Калифорния в Санта-Барбара за существенный конструктивный вклад в написание книги и значительную помощь.

Я благодарен доктору С. Амари (S. Amari) из Японии; докторам Сью Беккер (Sue Becker) и Рону Рэйсину (Ron Racine) из Макмастерского университета; доктору Син Холден (Sean Holden) из университетского колледжа в Лондоне; доктору Майклу Турмону (Michael Turmon), Пасадена; доктору Бабаку Хассиби (Babak Hassibi) из Стэндфордского университета; доктору Раулю Йии (Paul Yee), бывшему сотруднику Макмастерского университета, доктору Эдгару Осуна (Edgar Osuna) из Массачусетского технологического института; доктору Бернерду Шелкопфу (Bernard Schelkopf) из Института Макса Планка (Германия); доктору Майклу Джордану (Michael Jordan) из Массачусетского технологического института; докторам Редфорду Ниалу (Radford Neal) и Зубину Гархамани (Zoubin Gharhamani) из университета Торонто, доктору Джону Тситсиклису (John Tsitsiklis) из Массачусетского технологического института; доктору Джосу Принсипу (Jose Principe) из Университета штата Флорида; Гинту Пушкориусу (Gint Puskorius) и доктору Ли Фелдкампу (Lee Feldkamp) из исследовательской лаборатории Форда; доктору Ли Гилесу (Lee Giles) из исследовательского института компании NEC в Принстоне, штат Нью-Джерси; доктору Микелю Форкада (Mikel Forcada) из университета Далаканта (Испания); доктору Эрику Вану (Eric Wan) из института науки и технологии штата Орегона; а также доктору Яну Лекуну (Yann LeCun) из компании AT&T Research, штат Нью-Джерси, за их неоценимую помощь в улучшении данной книги.

Мне также хочется поблагодарить доктора Ральфа Линскера (Ralph Linsker) из компании IBM; доктора Стюарта Гемана (Stuart Geman) из Браунского университета; доктора Алана Гелфорда (Alan Gelford) из университета штата Коннектикут; доктора Барта Коско (Bart Kosko) из университета Южной Калифорнии; доктора Нариша Синха (Narish Sinha) из Макмастерского университета; доктора Костаса Диамантараса (Kostas Diamantaras) из университета г. Тессалоники имени Аристотеля (Греция); доктора Роберта Якобса (Robert Jacobs) из университета Рочестера; доктора Эндрю Барто (Andrew Barto) из Массачусетского университета; доктора Дона Хуша (Don Hush) из университета штата Нью-Мексико; доктора Йошуа Бенжио (Yoshua Bengio) из университета г. Монреаль; доктора Х. Янга (H. Yang) из института науки и технологий штата Орегона; доктора Скотта Дугласа (Scott Douglas) из университета штата Юта; доктора Натана Интратора (Nathan Intrator) из университета г. Тель-Авив (Израиль); доктора Владимира Вапника (Vladimir Vapnik) из компании AT&T Research, штат Нью-Джерси; доктора Тейво Кохонена (Teuvo Kohonen) из Хельсинского тех-

нологического университета (Финляндия); доктора Владимира Черкасского (Vladimir Cherkassky) из университета штата Миннесота; доктора Дэвида Лоува (David Lowe) из Астонского университета (Великобритания); доктора Джеймса Андерсона (James Anderson) из Браунского университета; доктора Андреаса Андрео (Andreas Andreou) из университета Джона Хопкина и доктора Томаса Анастасио (Thomas Anastasio) из университета штата Иллинойс.

Я глубоко благодарен моему студенту-дипломнику Хью Пасика (Hugh Pasika) за выполнение множества компьютерных экспериментов в среде MATLAB, а также за создание Web-узла этой книги. Большое спасибо также студенту-дипломнику Химешу Мадхуранату (Himesh Madhuranath); доктору Садасивану Путуссерипади (Sadasivan Puthusserypady); доктору Дж. Ни (J. Nie); доктору Паулю Йии (Paul Yee) и Гинту Пускориусу (Gint Puskorius) из лаборатории Ford Research за выполнение пяти экспериментов.

Большое спасибо также Хью Пасику (Hugh Pasika) за вычитку всей книги. В этой связи хочется также поблагодарить доктора Роберта Дони (Robert Dony), доктора Стефана Кремера (Stefan Kremer) и доктора Садасивана Путуссерипади (Sadasivan Puthusserypady) за вычитку отдельных глав книги.

Автор также выражает благодарность своему издателю Тому Роббинсу (Tom Robbins) и издателю Алис Дворкин (Alice Dworkin) за их полную поддержку и одобрение. Хочется поблагодарить Дженифер Маган (Jennifer Maughan) и сотрудников компании WestWorld, Inc. (штат Юта) за издание этой книги.

Автор выражает глубокую благодарность Бриджитт Майер (Brigitte Maier) из библиотеки Макмастерского университета за ее неоценимые усилия по поиску очень сложных ссылок и созданию столь полной библиографии. Следует отметить также помощь научного сотрудника библиотеки Пегги Финдлей (Peggy Findlay) и сотрудника библиотеки Регины Бендиг (Regina Bendig).

И наконец (но не в последнюю очередь), автор благодарен своему секретарю Лоле Брукс (Lola Brooks) за оформление различных версий рукописи. Без ее помощи написание этой книги и ее издание потребовали бы значительно больше времени.

Основные обозначения

Важные символы

a	Действие
$\mathbf{a}^T\mathbf{b}$	Скалярное произведение векторов **a** и **b**
$\mathbf{ab}^T$	Внешнее произведение векторов **a** и **b**
$\binom{l}{m}$	Биномиальный коэффициент
$A \cup B$	Объединение A и B
B	Величина, обратная температуре
b_k	Внешнее смещение, применяемое к нейрону k
cos(**a**, **b**)	Косинус угла между векторами **a** и **b**
D	Глубина памяти
$D_{f\|\|g}$	Дивергенция Куллбека–Леблера между функциями плотности вероятности f и g
$\tilde{\mathbf{D}}$	Оператор, сопряженный **D**
E	Функция энергии
E_i	Энергия состояния i в статистической механике
E	Статистический оператор ожидания
$\langle E \rangle$	Средняя энергия
erf	Функция ошибки
erfc	Дополнительная функция ошибки
exp	Экспонента
$\mathbf{E}_{av}$	Среднеквадратическая ошибка или сумма квадратических ошибок
$\mathbf{E}(n)$	Мгновенное значение суммы квадратических ошибок
F	Свободная энергия
$f_X(\mathbf{x})$	Функция плотности вероятности для случайного вектора **X**
$\mathbf{F}^*$	Подмножество (сетей) с минимальным эмпирическим риском
$\mathbf{H}$	Матрица Гессиана
$\mathbf{H}^{-1}$	Матрица, обратная Гессиану
i	Квадратный корень из -1, обозначаемый также как j
$\mathbf{I}$	Единичная матрица
$\mathbf{I}$	Информационная матрица Фишера
J	Среднеквадратическая ошибка
$\mathbf{J}$	Матрица Якобиана
$\mathbf{K}(n, n-1)$	Матрица ковариации ошибки в теории фильтрации Калмана

$\mathbf{K}^{1/2}$	Квадратный корень из матрицы $\mathbf{K}$
$\mathbf{K}^{T/2}$	Транспонированный квадратный корень из матрицы $\mathbf{K}$
k_B	Константа Больцмана
log	Логарифм
$L(\mathbf{w})$	Функция логарифмического правдоподобия от вектора весов $\mathbf{w}$
$\mathbf{L}(\mathbf{w})$	Функция логарифмического правдоподобия от вектора весов $\mathbf{w}$, построенная на основе одного примера
$\mathbf{M}_c$	Матрица управляемости
$\mathbf{M}_o$	Матрица наблюдаемости
n	Дискретное время
p_i	Вероятность состояния i в статистической механике
p_{ij}	Вероятность перехода из состояния i в состояние j
$\mathbf{P}$	Стохастическая матрица
P_c	Вероятность корректной классификации
P_e	Вероятность ошибки
$P(e\|\mathbf{C})$	Условная вероятность ошибки e при условии выбора входного воздействия из класса $\mathbf{C}$
p_α^+	Вероятность нахождения видимого нейрона машины Больцмана в состоянии α при условии фиксированной сети (положительная фаза)
p_α^-	Вероятность нахождения видимого нейрона машины Больцмана в состоянии α при условии свободного функционирования сети (отрицательная фаза)
$\hat{r}_x(j, k; n)$	Оценка автокорреляционной функции $x_j(n)$ и $x_k(n)$
$\hat{r}_{dx}(k; n)$	Оценка функции взаимной корреляции $d(n)$ и $x_k(n)$
$\mathbf{R}$	Матрица корреляции входного вектора
t	Непрерывное время
T	Температура
$\mathbf{T}$	Обучающее множество (выборка)
tr	След матрицы
var	Оператор дисперсии
$V(\mathbf{x})$	Функция Ляпунова от вектора состояния $\mathbf{x}$
v_j	Индуцированное локальное поле или активационный потенциал нейрона j
$\mathbf{w}_o$	Оптимальное значение вектора синаптических весов
w_{kj}	Синаптический вес синапса j, принадлежащего нейрону k
$\mathbf{w}^*$	Оптимальный вектор весов
$\bar{\mathbf{x}}$	Состояние равновесия вектора состояния $\mathbf{x}$
$\langle x_j \rangle$	Среднее значение состояния x_j в термальном смысле
$\hat{x}$	Оценка x, обозначенная символом “крышки”

$\|x\|$	Абсолютное значение (амплитуда) x
x^*	Комплексно сопряженное значение x, обозначенное звездочкой
$\|\|\mathbf{x}\|\|$	Евклидова норма (длина) вектора $\mathbf{x}$
$\mathbf{x}^T$	Вектор, транспонированный к $\mathbf{x}$
z^{-1}	Оператор единичной задержки
Z	Функция разбиения
$\delta_j(n)$	Локальный градиент нейрона j в момент времени n
Δw	Малое изменение вектора весов w
∇	Оператор градиента
∇^2	Оператор Лапласа
$\nabla_w J$	Градиент J по w
$\nabla \cdot \mathbf{F}$	Дивергенция вектора $\mathbf{F}$
η	Коэффициент скорости обучения
κ	Семиинвариант
μ	Стратегия
θ_k	Пороговое значение нейрона k (т.е. величина смещения b_k с обратным знаком)
λ	Параметр регуляризации
λ_k	k-е собственное значение квадратной матрицы
$\varphi_k(\cdot)$	Нелинейная функция активации нейрона k
$\in$	Символ принадлежности множеству
$\cup$	Символ объединения множеств
$\cap$	Символ пересечения множеств
$*$	Символ конволюции
$+$	Символ псевдообращения матрицы

Открытые и закрытые интервалы

Открытый интервал (a, b) изменения переменной x означает, что $a < x < b$.
Закрытый интервал $[a, b]$ изменения переменной x означает, что $a \leq x \leq b$.
Полуоткрытый интервал $[a, b)$ изменения переменной x означает, что $a \leq x < b$.
Аналогично, интервал $(a, b]$ изменения переменной x означает, что $a < x \leq b$.

Минимумы и максимумы

Символ $\arg\min_{\mathbf{w}} f(\mathbf{w})$ обозначает значение вектора аргумента $\mathbf{w}$, доставляющее минимум функции $f(\mathbf{w})$.

Символ $\arg\max_{\mathbf{w}} f(\mathbf{w})$ обозначает значение вектора аргумента $\mathbf{w}$, доставляющее максимум функции $f(\mathbf{w})$.

Введение

1.1. Что такое нейронные сети

Исследования по искусственным нейронным сетям (далее — нейронные сети) связаны с тем, что способ обработки информации человеческим мозгом в корне отличается от методов, применяемых обычными цифровыми компьютерами. Мозг представляет собой чрезвычайно *сложный*, *нелинейный*, *параллельный* компьютер (систему обработки информации). Он обладает способностью организовывать свои структурные компоненты, называемые *нейронами* (neuron), так, чтобы они могли выполнять конкретные задачи (такие как распознавание образов, обработку сигналов органов чувств, моторные функции) во много раз быстрее, чем могут позволить самые быстродействующие современные компьютеры. Примером такой задачи обработки информации может служить обычное *зрение* (human vision) [194], [637], [704]. В функции зрительной системы входит создание *представления* окружающего мира в таком виде, который обеспечивает возможность *взаимодействия* (interact) с этим миром. Более точно, мозг последовательно выполняет ряд задач распознавания (например, распознавание знакомого лица в незнакомом окружении). На это у него уходит около 100–200 миллисекунд, в то время как выполнение аналогичных задач даже меньшей сложности на компьютере может занять несколько дней.

Другим примером может служить *локатор* (sonar) летучей мыши, представляющий собой систему активной эхо-локации. Кроме предоставления информации о расстоянии до нужного объекта (например, мошки) этот локатор предоставляет информацию об относительной скорости объекта, о его размерах и размерах его отдельных элементов, а также об азимуте и высоте движения объекта [1026], [1027]. Для выделения этой информации из получаемого сигнала крохотный мозг летучей мыши проводит сложные нейронные вычисления. Эхо-локация летучей мыши по своим характеристикам качества и быстродействия превосходит самые сложные приборы, созданные инженерами.

Что же позволяет мозгу человека или летучей мыши добиться таких результатов? При рождении мозг имеет совершенную структуру, позволяющую строить собственные правила на основании того, что мы называем “опытом”. Опыт накапливается с

течением времени, и особенно масштабные изменения происходят в первые два года жизни человека. В этот период формируется остов общей структуры. Однако развитие на этом не прекращается — оно продолжается до последних дней жизни человека.

Понятие развития нейронов связано с понятием *пластичности* (plasticity) мозга — способности настройки нервной системы в соответствии с окружающими условиями. Именно пластичность играет самую важную роль в работе нейронов в качестве единиц обработки информации в человеческом мозге. Аналогично, в искусственных нейронных сетях работа проводится с искусственными нейронами. В общем случае *нейронная сеть* (neural network) представляет собой машину, моделирующую способ обработки мозгом конкретной задачи. Эта сеть обычно реализуется с помощью электронных компонентов или моделируется программой, выполняемой на цифровом компьютере. Предметом рассмотрения настоящей книги является важный класс нейронных сетей, осуществляющих вычисления с помощью процесса *обучения* (learning). Для того чтобы добиться высокой производительности, нейронные сети используют множество взаимосвязей между элементарными ячейками вычислений — *нейронами*. Таким образом, можно дать следующее определение нейронных сетей, выступающих в роли адаптивной машины[1].

> *Нейронная сеть — это громадный распределенный параллельный процессор, состоящий из элементарных единиц обработки информации, накапливающих экспериментальные знания и предоставляющих их для последующей обработки. Нейронная сеть сходна с мозгом с двух точек зрения.*
>
> - *Знания поступают в нейронную сеть из окружающей среды и используются в процессе обучения.*
> - *Для накопления знаний применяются связи между нейронами, называемые синаптическими весами.*

Процедура, используемая для процесса обучения, называется *алгоритмом обучения* (learning algorithm). Эта процедура выстраивает в определенном порядке синаптические веса нейронной сети для обеспечения необходимой структуры взаимосвязей нейронов.

Изменение синаптических весов представляет собой традиционный метод настройки нейронных сетей. Этот подход очень близок к теории линейных адаптивных фильтров, которая уже давно заявила о себе и применяется в различных областях деятельности человека [435], [1144]. Однако нейронные сети могут изменять собственную топологию. Это обусловлено тем фактом, что нейроны в человеческом мозге постоянно отмирают, а новые синаптические связи постоянно создаются.

---

[1] Это определение нейронных сетей взято из [16].

В литературе нейронные сети часто называют нейрокомпьютерами, сетями связей (connectionist network), параллельными распределенными процессорами и т.д. В этой книге мы будем пользоваться термином "нейронная сеть", хотя в отдельных случаях будут использоваться термины "нейрокомпьютер" и "сеть связей".

Преимущества нейронных сетей

Совершенно очевидно, что свою силу нейронные сети черпают, во-первых, из распараллеливания обработки информации и, во-вторых, из способности самообучаться, т.е. создавать обобщения. Под термином *обобщение* (generalization) понимается способность получать обоснованный результат на основании данных, которые не встречались в процессе обучения. Эти свойства позволяют нейронным сетям решать сложные (масштабные) задачи, которые на сегодняшний день считаются трудноразрешимыми. Однако на практике при автономной работе нейронные сети не могут обеспечить готовые решения. Их необходимо интегрировать в сложные системы. В частности, комплексную задачу можно *разбить* на последовательность относительно простых, часть из которых может решаться нейронными сетями. Очень важно уяснить, что для создания компьютерной архитектуры, которая будет способна имитировать человеческий мозг (если такое окажется возможным вообще), придется пройти долгий и трудный путь.

Использование нейронных сетей обеспечивает следующие полезные свойства систем.

1. *Нелинейность* (nonlinearity). Искусственные нейроны могут быть линейными и нелинейными. Нейронные сети, построенные из соединений нелинейных нейронов, сами являются нелинейными. Более того, эта нелинейность особого сорта, так как она *распределена* (distributed) по сети. Нелинейность является чрезвычайно важным свойством, особенно если сам физический механизм, отвечающий за формирование входного сигнала, тоже является нелинейным (например, человеческая речь).
2. *Отображение входной информации в выходную* (input-output mapping). Одной из популярных парадигм обучения является *обучение с учителем* (supervised learning). Это подразумевает изменение синаптических весов на основе набора маркированных *учебных примеров* (training sample). Каждый пример состоит из входного сигнала и соответствующего ему *желаемого отклика* (desired response). Из этого множества случайным образом выбирается пример, а нейронная сеть модифицирует синаптические веса для минимизации расхождений желаемого выходного сигнала и формируемого сетью согласно выбранному статистическому критерию. При этом собственно модифицируются *свободные параметры* (free parameters) сети. Ранее использованные примеры могут впоследствии быть применены снова,

но уже в другом порядке. Это обучение проводится до тех пор, пока изменения синаптических весов не станут незначительными. Таким образом, нейронная сеть обучается на примерах, составляя таблицу соответствий вход-выход для конкретной задачи. Такой подход заставляет вспомнить *непараметрическое статистическое обучение* (nonparametric statistical inference). Это направление статистики имеет дело с оценками, не связанными с какой-либо конкретной моделью, или, с биологической точки зрения, с обучением с нуля [344]. Здесь термин "непараметрический" используется для акцентирования того, что изначально не существует никакой предопределенной статистической модели входных данных. Для примера рассмотрим задачу *классификации образов* (pattern classification). В ней требуется соотнести входной сигнал, представляющий физический объект, или событие, с некоторой предопределенной категорией (классом). При непараметрическом подходе к этой задаче требуется "оценить" рамки решения в пространстве входного сигнала на основе набора примеров. При этом не используется никакая вероятностная модель распределения. Аналогичный подход применяется и в парадигме обучения с учителем. Это еще раз подчеркивает параллель между отображением входных сигналов в выходные, осуществляемым нейронной сетью, и непараметрическим статистическим обучением.

3. *Адаптивность* (adaptivity). Нейронные сети обладают способностью *адаптировать* свои синаптические веса к изменениям окружающей среды. В частности, нейронные сети, обученные действовать в определенной среде, могут быть легко переучены для работы в условиях незначительных колебаний параметров среды. Более того, для работы в *нестационарной* (nonstationary) среде (где статистика изменяется с течением времени) могут быть созданы нейронные сети, изменяющие синаптические веса в реальном времени. Естественная для классификации образов, обработки сигналов и задач управления архитектура нейронных сетей может быть объединена с их способностью к адаптации, что приведет к созданию моделей адаптивной классификации образов, адаптивной обработки сигналов и адаптивного управления. Известно, что чем выше адаптивные способности системы, тем более устойчивой будет ее работа в нестационарной среде. При этом хотелось бы заметить, что адаптивность не всегда ведет к устойчивости; иногда она приводит к совершенно противоположному результату. Например, адаптивная система с параметрами, быстро изменяющимися во времени, может также быстро реагировать и на посторонние возбуждения, что вызовет потерю производительности. Для того чтобы использовать все достоинства адаптивности, основные параметры системы должны быть достаточно стабильными, чтобы можно было не учитывать внешние помехи, и достаточно гибкими, чтобы обеспечить реакцию на существенные изменения среды. Эта задача обычно называется *дилеммой стабильности–пластичности* (stability-plasticity dilemma) [389].

4. *Очевидность ответа* (evidential response). В контексте задачи классификации образов можно разработать нейронную сеть, собирающую информацию не только для определения конкретного класса, но и для увеличения *достоверности* (confidence) принимаемого решения. Впоследствии эта информация может использоваться для исключения сомнительных решений, что повысит продуктивность нейронной сети.
5. *Контекстная информация* (contextual information). Знания представляются в самой структуре нейронной сети с помощью ее состояния активации. Каждый нейрон сети потенциально может быть подвержен влиянию всех остальных ее нейронов. Как следствие, существование нейронной сети непосредственно связано с контекстной информацией.
6. *Отказоустойчивость* (fault tolerance). Нейронные сети, облаченные в форму электроники, потенциально отказоустойчивы. Это значит, что при неблагоприятных условиях их производительность падает незначительно. Например, если поврежден какой-то нейрон или его связи, извлечение запомненной информации затрудняется. Однако, принимая в расчет распределенный характер хранения информации в нейронной сети, можно утверждать, что только серьезные повреждения структуры нейронной сети существенно повлияют на ее работоспособность. Поэтому снижение качества работы нейронной сети происходит медленно. Незначительное повреждение структуры никогда не вызывает катастрофических последствий. Это — очевидное преимущество робастных вычислений, однако его часто не принимают в расчет. Чтобы гарантировать отказоустойчивость работы нейронной сети, в алгоритмы обучения нужно закладывать соответствующие поправки [557].
7. *Масштабируемость* (VLSI Implementability). Параллельная структура нейронных сетей потенциально ускоряет решение некоторых задач и обеспечивает *масштабируемость* нейронных сетей в рамках технологии VLSI (very-large-scale-integrated). Одним из преимуществ технологий VLSI является возможность представить достаточно сложное поведение с помощью иерархической структуры [720].
8. *Единообразие анализа и проектирования* (Uniformity of analysis and design). Нейронные сети являются универсальным механизмом обработки информации. Это означает, что одно и то же проектное решение нейронной сети может использоваться во многих предметных областях. Это свойство проявляется несколькими способами.
   - Нейроны в той или иной форме являются стандартными составными частями *любой* нейронной сети.
   - Эта общность позволяет использовать одни и те же теории и алгоритмы обучения в различных нейросетевых приложениях.
   - Модульные сети могут быть построены на основе интеграции целых модулей.

9. *Аналогия с нейробиологией* (Neurobiological analogy). Строение нейронных сетей определяется аналогией с человеческим мозгом, который является живым доказательством того, что отказоустойчивые параллельные вычисления не только физически реализуемы, но и являются быстрым и мощным инструментом решения задач. Нейробиологи рассматривают искусственные нейронные сети как средство моделирования физических явлений. С другой стороны, инженеры постоянно пытаются почерпнуть у нейробиологов новые идеи, выходящие за рамки традиционных электросхем. Эти две точки зрения можно продемонстрировать на следующих примерах.

- В работе модели линейных систем вестибуло-окулярного рефлекса сравнивались с моделями *рекуррентных нейронных сетей* (они будут описаны в разделе 1.6 и более подробно в главе 15). *Вестибуло-окулярный рефлекс*, или рефлекс VOR (vestibulo-ocular reflex), является составной частью глазодвигательной системы. Его задачей является обеспечение стабильности визуального образа при поворотах головы за счет вращения глаз. Процесс VOR реализуется премоторными нейронами в вестибулярном центре, которые получают и обрабатывают сигналы поворота головы от вестибулярных сенсорных нейронов и передают результат на моторные нейроны мышц глаз. Механизм VOR хорошо подходит для моделирования, так как входной (поворот головы) и выходной (поворот глаз) сигналы можно точно описать. К тому же это довольно простой рефлекс, а нейрофизические свойства реализующих его нейронов довольно хорошо описаны. Среди трех задействованных в нем типов нейронов премоторные нейроны, входящие в состав вестибулярного центра, являются самыми сложными, а значит, самыми интересными. Ранее механизм VOR моделировался с помощью сосредоточенной линейной системы и теории управления. Эти модели были пригодны для описания некоторых общих свойств VOR, но не давали четкого представления о свойствах самих нейронов. С появлением нейросетевых моделей ситуация в корне изменилась. Рекуррентные модели VOR (программы, использующие алгоритм рекуррентного обучения в реальном времени, который будет описан в главе 15) позволили воспроизвести и описать многие статические, динамические, нелинейные и распределенные аспекты обработки сигналов при реализации рефлекса VOR и, в частности, вестибулярный центр [44].
- *Сетчатка* (retina), более чем какая-то другая часть мозга, выполняет функции взаимосвязи окружающего мира, представленного визуальным рядом или *физическим изображением* (physical image), проецируемым на матрицу рецепторов, с первым *нейронным изображением* (neural image). Сетчатка — это матрица микроскопических рецепторов на внешней лицевой стороне глазного яблока. В ее задачи входит преобразование оптического сигнала в нейронное изображение, передаваемое по оптическим нервам в различные центры для анализа. Принимая во внимание синаптическую организацию сетчатки, это — сложная задача. В любой сетчатке преобразование изображения из оптического в нейронное проходит три стадии [1017].

(i) Снятие фотокопии слоем нейронов-рецепторов.

(ii) Передача сформированного сигнала (реакция на свет) химическими синапсами на слой *биполярных клеток* (bipolar cell).

(iii) Передача этих сигналов (также с помощью химических синапсов) на выходные нейроны.

На двух последних стадиях (при передаче информации от рецепторов на биполярные рецепторы и от последних — на выходные нейроны) в операции участвуют специальные нейроны с латеральным торможением, в том числе так называемые *горизонтальные клетки* (horizontal cell). Их задачей является преобразование сигнала между разными синаптическими слоями. Также существуют *центробежные элементы*, обеспечивающие передачу сигнала с внутреннего синаптического слоя на внешний. Некоторые исследователи создавали электронные микросхемы, имитирующие структуру сетчатки [136], [137], [701]. Эти электронные чипы назывались *нейроморфными контурами* (neuromorphic integrated circuit) [720]. Нейроморфные сенсоры представляют собой матрицу фоторецепторов, связанных с соответствующими элементами рисунка (пикселями). Они имитируют сетчатку в том смысле, что могут адаптироваться к изменению освещенности, идентифицировать контуры и движение. Нейробиологическая модель, воплощенная в нейроморфные контуры, обеспечила еще одно преимущество: она вселила надежду на то, что физическое понимание нейробиологических структур может оказать существенное влияние на область электроники и технологию VLSI.

После рассмотрения этих примеров из нейробиологии становится ясно, почему столько внимания уделяется человеческому мозгу и его структурным уровням организации.

1.2. Человеческий мозг

Нервную систему человека можно рассматривать как трехступенчатую (рис. 1.1) [67]. Центром этой системы является *мозг* (brain), представленный *сетью нейронов* (*нервов*) (nerve net). Он получает информацию, анализирует ее и выдает соответствующие решения. На рис. 1.1 показаны два набора стрелок. Стрелки, направленные слева направо, обозначают *прямую* передачу сигналов информации в систему, а стрелки, направленные справа налево, — ответную реакцию системы. *Рецепторы* (receptor) преобразовывают сигналы от тела и из окружающей среды в электрические импульсы, передаваемые в нейронную сеть (мозг). *Эффекторы* (effector) преобразовывают электрические импульсы, сгенерированные нейронной сетью (мозгом), в выходные сигналы.

Рис. 1.1. Блочная диаграмма для нервной системы

Изучение человеческого мозга началось с работы, в которой предложена идея организации человеческого мозга на основе *нейронов* [869]. Как правило, реакция нейронов на 5-6 порядков медленнее реакции кремниевых логических элементов. Длительность событий в кремниевых элементах измеряется в наносекундах (10^{-9} с), а в нейронах — в миллисекундах (10^{-3} с). Однако эта относительная медлительность нейронов компенсируется их массой и количеством взаимосвязей между ними. По существующим оценкам, в коре головного мозга насчитывается около 10 биллионов нейронов и около 60 триллионов синапсов или взаимосвязей между нейронами [977]. В результате мозг представляет собой чрезвычайно *эффективную структуру*. В частности, энергетические затраты мозга на выполнение одной операции в секунду составляют около 10^{-16} Дж. В то же время затраты самого экономичного компьютера не опускаются ниже 10^{-6} Дж на операцию в секунду [286].

Синапсы (synapses) — это элементарные структурные и функциональные единицы, которые передают импульсы между нейронами. Самым распространенным типом синапсов являются *химические* (chemical synapse), которые работают следующим образом. Предсинаптический процесс формирует *передаваемую субстанцию* (transmitter substance), которая методом диффузии передается по синаптическим соединениям между нейронами и влияет на постсинаптический процесс. Таким образом, синапс преобразовывает предсинаптический электрический сигнал в химический, а после этого — в постсинаптический электрический [977]. В электротехнической терминологии его можно было бы назвать *невзаимным четырехполюсником* (nonreciprocal two-port device). В традиционных описаниях нейронной организации синапс представляют простым соединением, которое может передавать *возбуждение* (excitation) или *торможение* (inhibition) (но не то и другое одновременно) между нейронами.

Ранее уже говорилось, что под пластичностью понимается способность нервной системы адаптироваться к условиям окружающей среды [194], [279]. В мозге взрослого человека пластичность реализуется двумя механизмами: путем создания новых синаптических связей между нейронами и за счет модификации существующих. *Аксоны* (axon) (линии передачи) и *дендриты* (dendrite) (зоны приема) представляют собой два типа элементов клетки, которые различаются даже на морфологическом уровне. Аксоны имеют более гладкую поверхность, более тонкие границы и большую длину. Дендриты (свое название они получили из-за сходства с деревом) имеют неровную поверхность с множеством окончаний [315]. Существует огромное множество форм и размеров нейронов, в зависимости от того, в какой части мозга они находятся. На рис. 1.2 показана *пирамидальная клетка* (pyramidal cell) — самый распространен-

ный тип нейронов коры головного мозга. Как и все нейроны, пирамидальные клетки получают сигналы от щупалец дендритов (фрагмент дендрита показан на рис. 1.2). Пирамидальная клетка может получать более 10000 синаптических сигналов и проектировать их на тысячи других клеток.

Выходные сигналы большинства нейронов преобразуются в последовательность коротких электрических импульсов. Эти импульсы, называемые *потенциалами действия* (action potential) или *выбросами* (spike), берут свое начало в теле нейрона и передаются через другие нейроны с постоянной скоростью и амплитудой. Причина использования потенциала действия для взаимодействия нейронов лежит в самой физической природе аксона. Аксон нейрона имеет большую длину и маленькую толщину, что выражается в его большом электрическом сопротивлении и емкости. Обе эти характеристики распределены по аксону. Таким образом, его можно смоделировать как линию электропередачи с использованием *уравнения кабеля* (cable equation). Анализ этого уравнения показывает, что подаваемое на один конец аксона напряжение экспоненциально уменьшается с расстоянием, достигая на другом его конце малых значений. Потенциалы действия позволяют обойти эту проблему [46].

В человеческом мозге существуют крупно- и мелкомасштабные анатомические структуры. Эти верхний и нижний уровни отвечают за выполнение разных функций. На рис. 1.3 показана иерархия уровней организации мозга, составленная на основе анализа отдельных областей мозга [194], [977]. *Синапсы* (synapse) представляют собой самый нижний уровень — уровень молекул и ионов. На следующих уровнях мы имеем дело с нейронными микроконтурами, дендритными деревьями и в завершение — с нейронами. Под *нейронным микроконтуром* (neural microcircuit) понимается набор синапсов, организованный в шаблоны взаимосвязей, выполняющих определенную операцию. Нейронный микроконтур можно сравнить с электронным чипом, состоящим из набора транзисторов. Минимальный размер микроконтуров измеряется в микронах, а скорость операций — в миллисекундах. Нейронные микроконтуры группируются в *дендритные субблоки* (dendritic subunit), составляющие *дендритные деревья* (dendritic tree) отдельных нейронов. Весь *нейрон* (neuron) имеет размеры порядка 100 микрон и содержит несколько дендритных субблоков. На следующем уровне сложности находятся *локальные цепочки* (local circuit) (размером порядка 1 мм), состоящие из нейронов с одинаковыми или сходными характеристиками. Эти наборы нейронов выполняют функции, характерные для отдельных областей мозга. За ними в иерархии следуют *межрегиональные цепочки* (interregional circuit), состоящие из траекторий, столбцов и топографических карт и объединяющие несколько областей, находящихся в разных частях мозга.

Топографические карты (topographic map) предназначены для ответа на поступающую от сенсоров информацию. Эти карты часто организуются в виде таблиц, причем визуальные, звуковые и соматосенсорные карты хранятся в стеке, сохраняющем пространственную конфигурацию конкретных точек возбуждения. На рис. 1.4

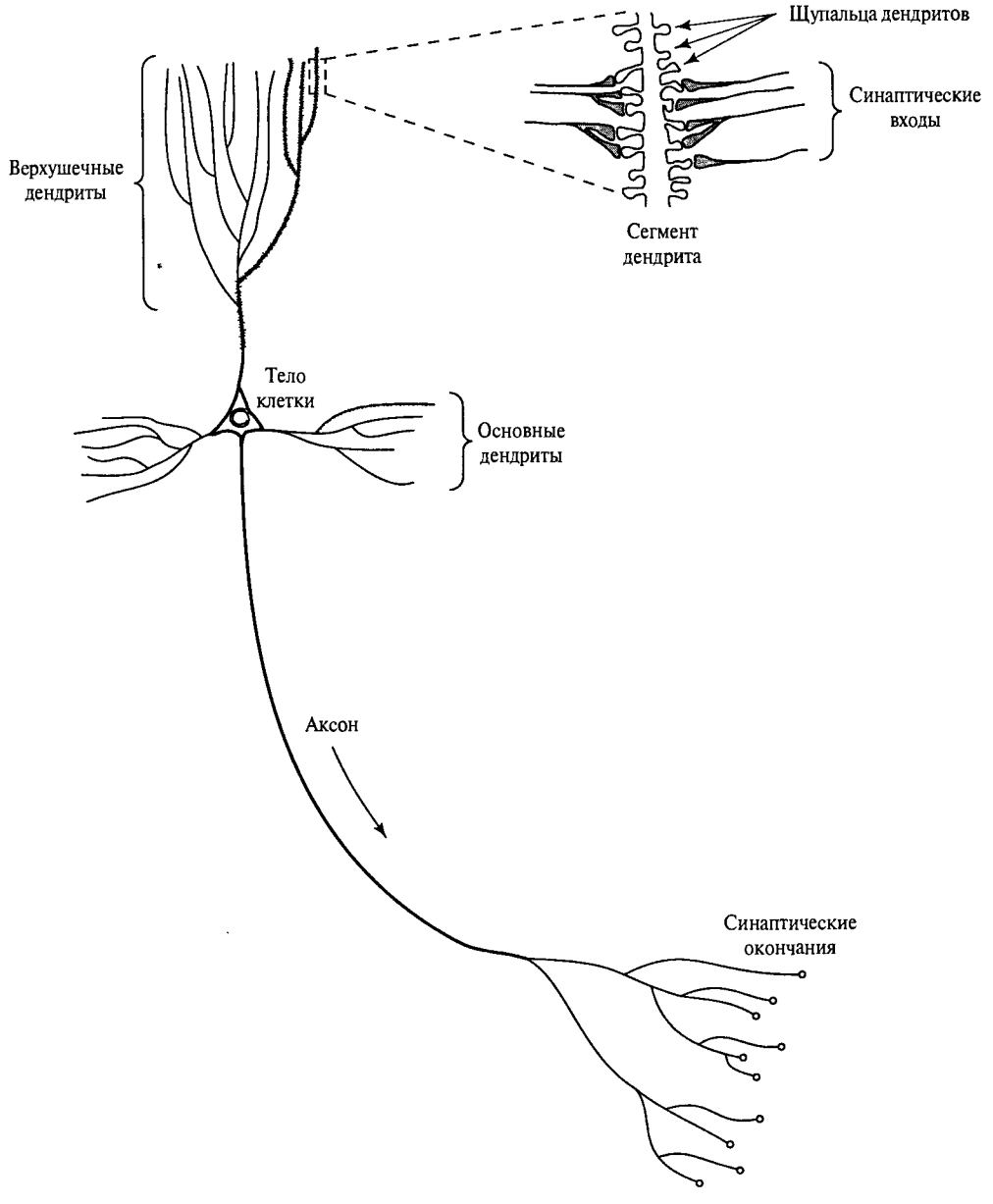


Рис. 1.2. Пирамидальная клетка

показана цитоархитектурная карта церебральной коры мозга, разработанная в [157]. На этом рисунке четко видно, что разные сенсорные сигналы (моторные, соматические, визуальные и т.п.) отображаются на соответствующие области церебральной коры с сохранением порядка. На заключительном уровне сложности топографические карты и прочие межрегиональные цепочки связываются друг с другом, образуя *центральную нервную систему* (central nervous system).

Рис. 1.3. Структурная организация уровней мозга

Очень важно уяснить, что описанные структурные уровни организации являются уникальными характеристиками мозга. Здесь ничто не напоминает цифровой компьютер, и в работе с нейронными сетями мы нигде с ним не столкнемся. Тем не менее мы направим свои стопы в сторону создания иерархии вычислительных уровней, напоминающих представленные на рис. 1.3. Искусственные нейроны, которые мы будем использовать для создания нейронных сетей, сильно упрощены по сравнению с их биологическими прототипами. Нейронные сети, которые можно создать в настоящее время, примитивны по сравнению с локальными и межрегиональными цепочками мозга. Утешением служит только тот факт, что за последние два десятилетия произошел большой прорыв на многих фронтах исследований. Принимая во внимание аналогию с нейробиологией, а также многообразие теоретических и технологических средств, можно предположить, что следующее десятилетие принесет более глубокое понимание искусственных нейронных сетей.

Предметом настоящей книги является изучение нейронных сетей с точки зрения инженерии[2]. Сначала будут описаны модели искусственных нейронов, формирующих основу нейронных сетей. Сами нейронные сети рассматриваются в последующих главах.

[2] Применение нейронных сетей в области нейронного моделирования, познания и нейрофизиологии описано в [46]. Понятное описание вычислительных аспектов мозга содержнтся в [194]. Детальное описание нейронных механизмов и человеческого мозга содержится в [315], [536], [565], [604], [975], [976].

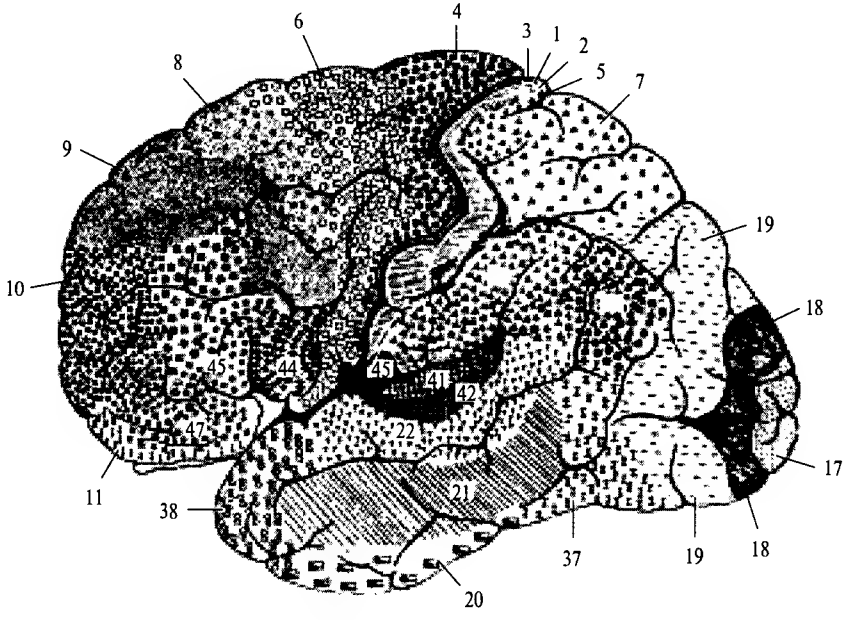


Рис. 1.4. Цитоархитектурная карта церебральной коры мозга. Различные области отличаются между собой по толщине слоя и типам составляющих их клеток. Выделим некоторые наиболее важные области. К моторной коре относятся область 4 — моторная цепочка, премоторная область 6; фронтальная область зрения 8. Соматосенсорную кору составляют области 3, 1, 2. Области 17–19 относятся к визуальной коре, а области 41 и 42 — к слуховой. (На основании [157], с разрешения издательства Oxford University Press.)

1.3. Модели нейронов

Нейрон представляет собой единицу обработки информации в нейронной сети. На блок-схеме рис. 1.5 показана *модель* (model) нейрона, лежащего в основе искусственных нейронных сетей. В этой модели можно выделить три основных элемента.

1. Набор *синапсов* (synapse) или *связей* (connecting link), каждый из которых характеризуется своим *весом* (weight) или *силой* (strength). В частности, сигнал x_j на входе синапса j, связанного с нейроном k, умножается на вес w_{kj}. Важно обратить внимание на то, в каком порядке указаны индексы синаптического веса w_{kj}. Первый индекс относится к рассматриваемому нейрону, а второй — ко входному окончанию синапса, с которым связан данный вес. В отличие от синапсов мозга синаптический вес искусственного нейрона может иметь как положительные, так и отрицательные значения.
2. *Сумматор* (adder) складывает входные сигналы, взвешенные относительно соответствующих синапсов нейрона. Эту операцию можно описать как *линейную комбинацию*.

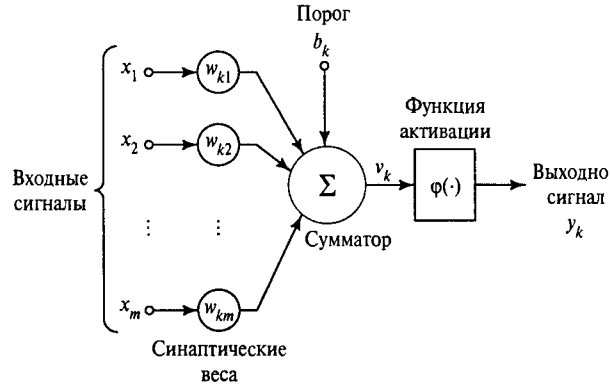


Рис. 1.5. Нелинейная модель нейрона

3. *Функция активации* (activation function) ограничивает амплитуду выходного сигнала нейрона. Эта функция также называется функцией *сжатия* (squashing function). Обычно нормализованный диапазон амплитуд выхода нейрона лежит в интервале $[0, 1]$ или $[-1, 1]$.

В модель нейрона, показанную на рис. 1.5, включен *пороговый элемент* (bias), который обозначен символом b_k. Эта величина отражает увеличение или уменьшение входного сигнала, подаваемого на функцию активации.

В математическом представлении функционирование нейрона k можно описать следующей парой уравнений:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j, \tag{1.1}$$

$$y_k = \varphi(u_k + b_k) \tag{1.2}$$

где $x_1, x_2, \ldots, x_m$ — входные сигналы; $w_{k1}, w_{k2}, \ldots, w_{km}$ — синаптические веса нейрона k; u_k — *линейная комбинация входных воздействий* (linear combiner output); b_k — порог; $\varphi(\cdot)$ — *функция активации* (activation function); y_k — выходной сигнал нейрона. Использование порога b_k обеспечивает эффект *афинного преобразования* (affine transformation) выхода линейного сумматора u_k. В модели, показанной на рис. 1.5, постсинаптический потенциал вычисляется следующим образом:

$$v_k = u_k + b_k. \tag{1.3}$$

В частности, в зависимости от того, какое значение принимает порог b_k, положительное или отрицательное, *индуцированное локальное поле* (induced local field) или *потенциал активации* (activation potential) v_k нейрона k изменяется так, как показано

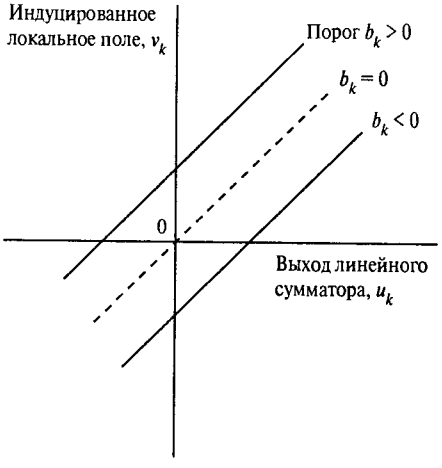


Рис. 1.6. Афинное преобразование, вызванное наличием порога. Обратите внимание, что в точке, где $u_k = 0, v_k = b_k$

на рис. 1.6. Здесь и далее мы будем использовать термин "индуцированное локальное поле". Обратите внимание на результат этого афинного преобразования. График v_k уже не проходит через начало координат, как график u_k.

Порог b_k является внешним параметром искусственного нейрона k. Его присутствие мы видим в выражении (1.2). Принимая во внимание выражение (1.3), формулы (1.1), (1.2) можно преобразовать к следующему виду:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j, \tag{1.4}$$

$$y_k = \varphi(v_k) \tag{1.5}$$

В выражении (1.4) добавился новый синапс. Его входной сигнал равен:

$$x_0 = +1, \tag{1.6}$$

а его вес:

$$w_{k0} = b_k. \tag{1.7}$$

Это позволило трансформировать модель нейрона к виду, показанному на рис. 1.7. На этом рисунке видно, что в результате введения порога добавляется новый входной сигнал фиксированной величины $+1$, а также появляется новый синаптический вес, равный пороговому значению b_k. Хотя модели, показанные на рис. 1.5 и 1.7, внешне совершенно не схожи, математически они эквивалентны.

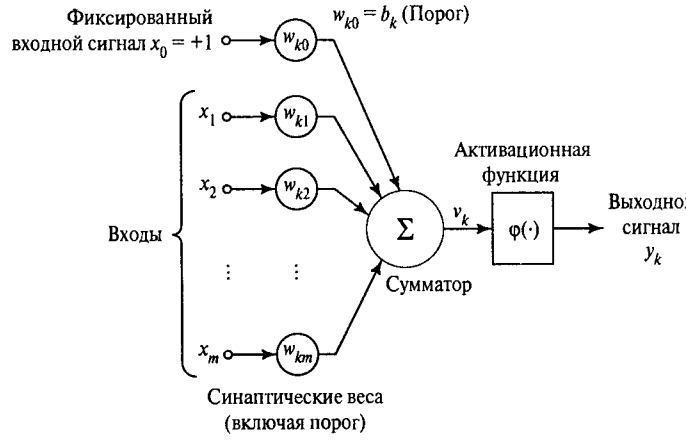


Рис. 1.7. Еще одна нелинейная модель нейрона

Типы функций активации

Функции активации, представленные в формулах как $\varphi(v)$, определяют выходной сигнал нейрона в зависимости от индуцированного локального поля v. Можно выделить три основных типа функций активации.

1. *Функция единичного скачка*, или пороговая функция (threshold function). Этот тип функции показан на рис. 1.8, *а* и описывается следующим образом:

$$\varphi(v) = \begin{cases} 1, \text{ если } v \geq 0; \\ 0, \text{ если } v < 0; \end{cases} \tag{1.8}$$

В технической литературе эта форма функции единичного скачка обычно называется *функцией Хэвисайда* (Heaviside function). Соответственно выходной сигнал нейрона k такой функции можно представить как

$$y_k = \begin{cases} 1, \text{ если } v_k \geq 0; \\ 0, \text{ если } v_k < 0; \end{cases} \tag{1.9}$$

где v_k — это индуцированное локальное поле нейрона, т.е.

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k. \tag{1.10}$$

Эту модель в литературе называют *моделью Мак-Каллока–Питца* (McCalloch-Pitts model), отдавая дань пионерской работе [714]. В этой модели выходной сигнал нейрона принимает значение 1, если индуцированное локальное поле этого нейрона не отрицательно, и 0 — в противном случае. Это выражение описывает свойство *"все или ничего"* модели Мак-Каллока–Питца.

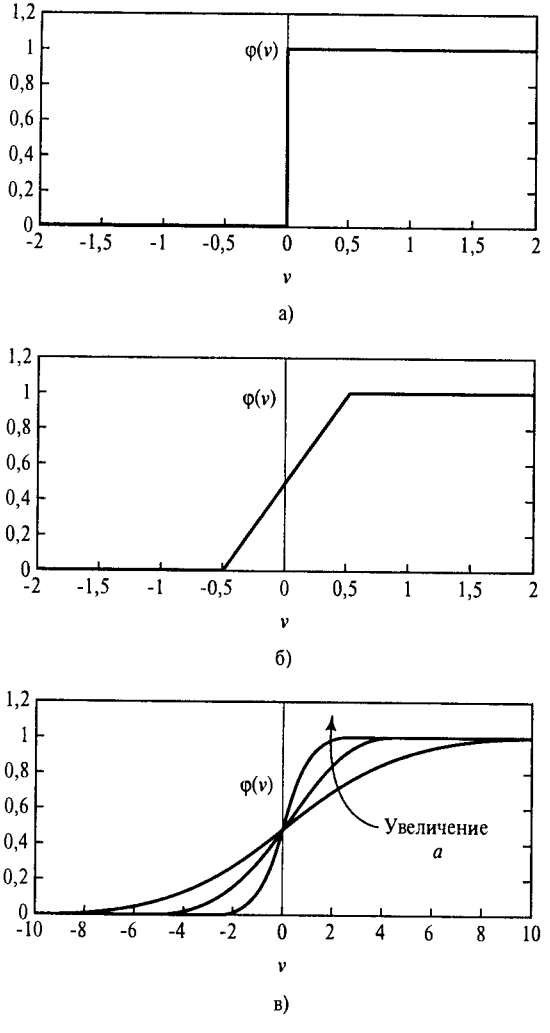


Рис. 1.8. Виды активационных функций: функция единичного скачка (а); кусочно-линейная функция (б) и сигмоидальная функция для различных значений параметра *a* (в)

2. *Кусочно-линейная функция* (piecewise-linear function). Кусочно-линейная функция, показанная на рис. 1.8, *б*, описывается следующим выражением:

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2}; \\ |v|, & +\frac{1}{2} > v > -\frac{1}{2}; \\ 0, & v \leq -\frac{1}{2}, \end{cases} \tag{1.11}$$

где коэффициент усиления в линейной области оператора предполагается равным единице. Эту функцию активации можно рассматривать как *аппроксимацию* (approximation) нелинейного усилителя. Следующие два варианта можно считать особой формой кусочно-линейной функции.

- Если линейная область оператора не достигает порога насыщения, он превращается в *линейный сумматор* (linear combiner).
- Если коэффициент усиления линейной области принять бесконечно большим, то кусочно-линейная функция вырождается в *пороговую* (threshold function).

3. *Сигмоидальная функция* (sigmoid function). Сигмоидальная функция, график которой напоминает букву S, является, пожалуй, самой распространенной функцией, используемой для создания искусственных нейронных сетей. Это быстро возрастающая функция, которая поддерживает баланс между линейным и нелинейным поведением[3]. Примером сигмоидальной функции может служить *логистическая функция*[4] (logistic function), задаваемая следующим выражением:

$$\varphi(v) = \frac{1}{1 + \exp(-av)}, \tag{1.12}$$

где a — *параметр наклона* (slope parameter) сигмоидальной функции. Изменяя этот параметр, можно построить функции с различной крутизной (см. рис. 1.8, *в*). Первый график соответствует величине параметра, равной $a/4$. В пределе, когда параметр наклона достигает бесконечности, сигмоидальная функция вырождается в пороговую. Если пороговая функция может принимать только значения 0 и 1, то сигмоидальная функция принимает бесконечное множество значений в диапазоне от 0 до 1. При этом следует заметить, что сигмоидальная функция является дифференцируемой, в то время как пороговая — нет. (Как мы увидим в главе 4, дифференцируемость активационной функции играет важную роль в теории нейронных сетей.)

Область значений функций активации, определенных формулами (1.8), (1.11) и (1.12), представляет собой отрезок от 0 до $+1$. Однако иногда требуется функция активации, имеющая область значений от -1 до $+1$. В этом случае функция активации должна быть симметричной относительно начала координат. Это значит, что функция активации является нечетной функцией индуцированного локального поля. В частности, пороговую функцию в данном случае можно определить следующим образом:

$$\varphi(v) = \begin{cases} 1, & \text{если } v > 0; \\ 0, & \text{если } v = 0; \\ -1, & \text{если } v < 0, \end{cases} \tag{1.13}$$

---

[3] Полное описание сигмоидальной функции и связанных с ней вопросов содержится в [727].

[4] Логистическая функция, или, более точно, функция логистического распределения, описывает широко представленный в литературе закон логистического роста. Все процессы роста могут быть представлены функцией логистического распределения $F(t) = 1/(1+\exp(\alpha t - \beta))$, где t — переменная времени; α и β — константы. Следует отметить, что с таким же или даже с большим успехом к тем же данным можно применить и другие распределения, например распределение Гаусса [294].

Эта функция обычно называется *сигнум*. В данном случае сигмоидальная функция будет иметь форму *гиперболического тангенса*:

$$\varphi(v) = \tanh(v). \tag{1.14}$$

Такой вид сигмоидальной функции обеспечивает ряд преимуществ, о которых речь пойдет в главе 4.

Стохастическая модель нейрона

Модель нейрона, показанная на рис. 1.7, является детерминистской. Это значит, что преобразование входного сигнала в выходной точно определено для всех значений входного сигнала. Однако в некоторых приложениях лучше использовать стохастические нейросетевые модели, в которых функция активации имеет вероятностную интерпретацию. В подобных моделях нейрон может находиться в одном из двух состояний: $+1$ или -1. Решение о переключении состояния нейрона принимается с учетом вероятности этого события. Обозначим состояние нейрона символом x, а *вероятность активации нейрона* (probability of firing) — функцией $P(v)$, где v — индуцированное локальное поле нейрона. Тогда

$$x = \begin{cases} +1, & \text{с вероятностью } P(v); \\ -1, & \text{с вероятностью } 1 - P(v). \end{cases}$$

Вероятность $P(v)$ описывается сигмоидальной функцией следующего вида [660]:

$$P(v) = \frac{1}{1 + \exp(-v/T)}, \tag{1.15}$$

где T — это аналог *температуры* (temperature), используемый для управления уровнем шума, и, таким образом, степенью неопределенности переключения. При этом важно заметить, что T *не описывает* физическую температуру нейронной сети, будь то биологической или искусственной. Параметр T управляет термальными флуктуациями, представляющими эффект синаптического шума. Заметим, что если параметр T стремится к нулю, то стохастический нейрон, описанный выражением (1.15), принимает детерминированную форму (без включения шума) нейрона Мак-Каллока–Питца.

1.4. Представление нейронных сетей с помощью направленных графов

Блочные диаграммы (block diagram), представленные на рис. 1.5 и 1.7, обеспечивают функциональное описание различных элементов, из которых состоит модель искусственного нейрона. Внешний вид модели можно в значительной мере упростить, применив идею графов прохождения сигнала. Графы передачи сигнала (signal-flow) с наборами правил были введены Мейсоном (Mason) в 1953 году для описания линейных сетей [706], [707]. Поэтому наличие нелинейности в модели нейрона ограничивает область применения этой парадигмы в нейронных сетях. Тем не менее графы прохождения сигнала дают хорошее представление о передаче сигнала по нейронным сетям. Именно этот вопрос и будет рассматриваться в данном разделе.

Граф передачи (или прохождения) сигнала (signal-flow graph) — это сеть направленных *связей* (links) (или *ветвей* (branches)), соединяющих отдельные точки (*узлы*). С каждым узлом j связан *сигнал* x_j. Обычная направленная связь начинается в некотором узле j и заканчивается в другом узле k. С ней связана некоторая *передаточная функция* (transfer function), определяющая зависимость сигнала y_k узла k от сигнала x_j узла j. Прохождение сигнала по различным частям графа подчиняется трем основным правилам.

Правило 1. Направление прохождения сигнала вдоль каждой связи определяется направлением стрелки. При этом можно выделить два типа связей.

- *Синаптические связи* (synaptic link). Их поведение определяется *линейным* (linear) соотношением вход-выход. А именно, как показано на рис. 1.9, *а*, сигнал узла x_j умножается на синаптический вес w_{kj}, в результате чего получается сигнал узла y_k.
- *Активационные связи* (activation link). Их поведение определяется *нелинейным* (nonlinear) соотношением вход-выход. Этот вид связи показан на рис. 1.9, *б*, где $\varphi(\cdot)$ — нелинейная функция активации.

Правило 2. Сигнал узла равен алгебраической сумме сигналов, поступающих на его вход. На рис. 1.9, *в* это правило проиллюстрировано для случая *синаптической сходимости* (synaptic convergence).

Правило 3. Сигнал данного узла передается по каждой исходящей связи без учета передаточных функций исходящих связей. Это правило проиллюстрировано на рис. 1.9, *г* для *синаптической дивергенции* (synaptic divergence) или расходимости.

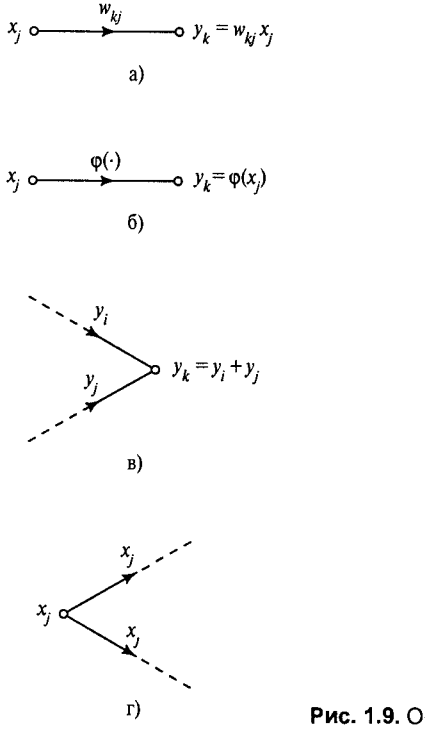


Рис. 1.9. Основные правила построения графов передачи сигналов

На рис. 1.10 показан пример графа передачи сигнала. Это модель нейрона, соответствующая блочной диаграмме, приведенной на рис. 1.7. По своему внешнему виду граф на рис. 1.10 значительно проще диаграммы на рис. 1.7, хотя и содержит все функциональные детали. Обратите внимание, что на обоих рисунках входной сигнал принимает значение $x_0 = +1$, а соответствующий синаптический вес $w_{k0} = b_k$, где b_k — порог нейрона k.

Принимая в качестве модели нейрона граф прохождения сигнала, показанный на рис. 1.10, можно сформулировать еще одно определение нейронной сети.

Нейронная сеть — это направленный граф, состоящий из узлов, соединенных синаптическими и активационными связями, который характеризуется следующими четырьмя свойствами.

1. *Каждый нейрон представляется множеством линейных синаптических связей, внешним порогом и, возможно, нелинейной связью активации. Порог, представляемый входной синаптической связью, считается равным $+1$.*
2. *Синаптические связи нейрона используются для взвешивания соответствующих входных сигналов.*
3. *Взвешенная сумма входных сигналов определяет индуцированное локальное поле каждого конкретного нейрона.*
4. *Активационные связи модифицируют индуцированное локальное поле нейрона, создавая выходной сигнал.*

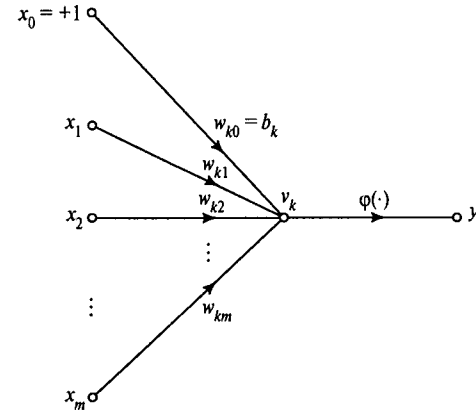


Рис. 1.10. Граф передачи сигнала для одного нейрона

Состояние нейрона может быть определено в терминах индуцированного локального поля или его выходного сигнала.

Направленный граф, определенный указанным выше способом, является *полным* (complete). Это значит, что он описывает не только прохождение сигнала между нейронами, но и передачу сигнала в самом нейроне. Если необходимо описать только прохождение сигнала между нейронами, то можно использовать сокращенную форму этого графа, опускающую детали передачи сигнала внутри нейронов. Такой направленный граф называется *частично полным* (partially complete) и характеризуется следующими свойствами.

1. Входные сигналы графа формируются *узлами источника* (source node) или входными элементами.
2. Каждый нейрон представляется одним узлом, который называется *вычислительным* (computation node).
3. *Линии передачи сигнала* (communication links), соединяющие узлы-источники и вычислительные узлы графа, не имеют веса. Они просто определяют направление прохождения сигнала на графе.

Частично полный направленный граф, определенный указанным выше способом, называется *архитектурным* (architectural graph). Он описывает структуру нейронной сети. Простейший случай графа, состоящего из одного нейрона с m входами и фиксированный порогом, равным $+1$, показан на рис. 1.11. Обратите внимание, что на этом графе вычислительный узел обозначен заштрихованным кружком, а узлы источника — маленькими квадратиками. Это соглашение будет использоваться на протяжении всей книги. Более сложные примеры нейросетевой архитектуры приводятся в разделе 1.6.

Подводя итог сказанному, можно выделить три графических представления нейронных сетей.

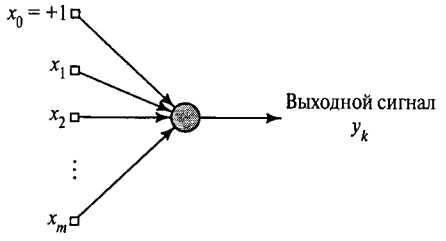


Рис. 1.11. Архитектурный граф нейрона

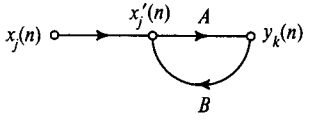


Рис. 1.12. Граф передачи сигнала в системе с одной обратной связью

- Блочная диаграмма, описывающая функции нейронной сети.
- Граф прохождения сигнала, обеспечивающий полное описание передачи сигнала по нейронной сети.
- Архитектурный граф, описывающий структуру нейронной сети.

1.5. Обратная связь

Понятие *обратной связи* (feedback) характерно для динамических систем, в которых выходной сигнал некоторого элемента системы оказывает влияние на входной сигнала этого элемента. Таким образом, некоторые внешние сигналы усиливаются сигналами, циркулирующими внутри системы. На самом деле обратная связь присутствует в нервной системе практически любого животного [315]. Более того, она играет важную роль в изучении особого класса нейронных сетей, называемых *рекуррентными* (recurrent network). На рис. 1.12 показан граф прохождения сигнала в *системе с одной обратной связью* (single-loop feedback system). В ней входной сигнал $x_j(n)$, внутренний сигнал $x'_j(n)$ и выходной сигнал $y_j(n)$ являются функциями дискретной переменной n. Предполагается, что эта система линейна и содержит прямую и обратную связи, которые характеризуются операторами A и B соответственно. В частности, выходной сигнал прямого канала частично определяет значение канала обратной связи. Из рис. 1.12 прослеживается следующая зависимость:

$$y_k(n) = A[x'_j(n)], \tag{1.16}$$

$$x'_j(n) = x_j(n) + B[y_j(n)], \tag{1.17}$$

где квадратные скобки обозначают операторы A и B.

Исключая переменную $x'_j(n)$ в выражениях (1.16) и (1.17), получим:

$$y_k(n) = \frac{A}{1 - AB}[x_j(n)]. \tag{1.18}$$

Выражение $A/(1 - AB)$ называют *оператором замкнутого контура* (closed-loop operator) системы, а выражение AB — *оператором разомкнутого контура* (open-loop operator). В общем случае оператор разомкнутого контура не обладает свойством коммутативности, т.е. $BA \neq AB$.

Для примера рассмотрим систему с одной обратной связью, показанную на рис. 1.13. В ней предполагается, что оператор A — это фиксированный вес w, а B является *оператором единичной задержки* (unit-delay operator) z^{-1}, который задерживает выходной сигнал по отношению к входному на один шаг дискретизации. Исходя из этого, оператор замкнутого контура можно представить следующим образом:

$$\frac{A}{1 - AB} = \frac{w}{1 - wz^{-1}} = w(1 - wz^{-1})^{-1}.$$

Используя биномиальное представление выражения $(1 - wz^{-1})^{-1}$, оператор замкнутого контура можно записать в виде

$$\frac{A}{1 - AB} = w \sum_{l=0}^{\infty} w^l z^{-l}. \tag{1.19}$$

Подставляя выражение (1.19) в (1.18), получим:

$$y_k(n) = w \sum_{l=0}^{\infty} w^l z^{-l}[x_j(n)]. \tag{1.20}$$

Здесь квадратные скобки снова обозначают тот факт, что z^{-1} является оператором. Из определения этого оператора имеем

$$z^{-l}[x_j(n)] = x_j(n - l), \tag{1.21}$$

где $x_j(n - l)$ — это входной сигнал, задержанный на l единиц дискретизации. Следовательно, выходной сигнал $y_k(n)$ можно представить как бесконечную взвешенную сумму текущего и предыдущих значений входного сигнала $x_j(n)$:

$$y_k(n) = \sum_{l=0}^{\infty} w^{l+1} x_j(n - l). \tag{1.22}$$

На рис. 1.14 ясно видно, что динамика реакции системы определяется весом w. Здесь можно выделить два случая.

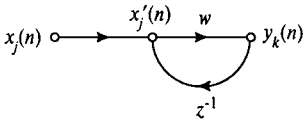


Рис. 1.13. Граф прохождения сигнала для фильтра с бесконечной импульсной характеристикой (IIR-filter — infinite-duration impulse response filter) первого порядка

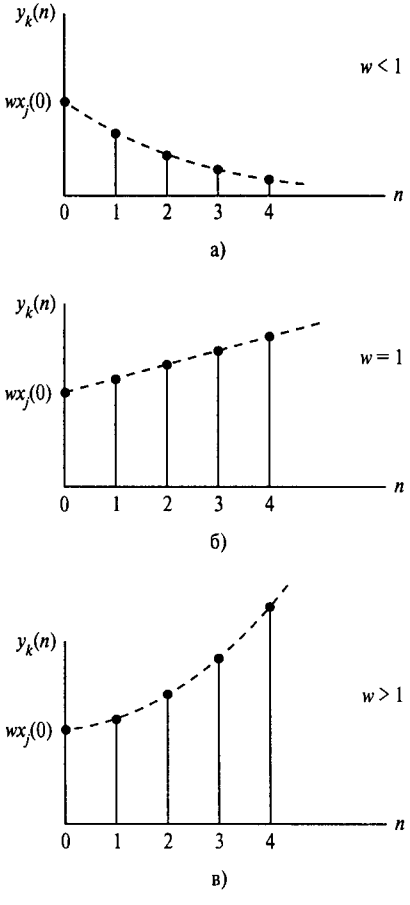


Рис. 1.14. Реакция системы, показанной на рис. 1.13, для трех различных значений веса w: устойчивая система (а), случай линейной (б) и экспоненциальной расходимости (в)

1. $|w| < 1$. В данном случае выходной сигнал $y_k(n)$ экспоненциально *сходится* (convergent). Это значит, что система *устойчива* (stable). Для положительного значения w такая реакция показана на рис. 1.14, *а*.
2. $|w| \geq 1$. В данном случае выходной сигнал $y_k(n)$ *расходится* (divergent). Это значит, что система *неустойчива* (unstable). Если $|w| = 1$, расходимость является линейной (рис. 1.14, *б*); если $|w| > 1$ — экспоненциальной (рис. 1.14, *в*).

Устойчивость — одна из основных характеристик, которой уделяется большое внимание при изучении систем с обратной связью.

Случай $|w| < 1$ соответствует системам с бесконечной памятью, в том смысле, что выход системы зависит от состояния системы в бесконечно далеком прошлом. Это влияние *вырождается* (fading), т.е. экспоненциально уменьшается с течением времени.

Анализ динамики поведения нейронных сетей с обратной связью усложняется тем, что процессорные элементы сети обычно *нелинейны* (nonlinear). Такие нейронные сети будут рассматриваться в последней части данной книги.

1.6. Архитектура сетей

Структура нейронных сетей тесно связана с используемыми алгоритмами обучения. Классификация алгоритмов обучения будет приведена в следующей главе, а вопросы их построения будут изучены в последующих главах. В данном разделе мы сосредоточим внимание на архитектурах сетей (структурах).

В общем случае можно выделить три фундаментальных класса нейросетевых архитектур.

Однослойные сети прямого распространения

В *многослойной* нейронной сети нейроны располагаются по слоям. В простейшем случае в такой сети существует *входной слой* (input layer) узлов источника, информация от которого передается на *выходной слой* (output layer) нейронов (вычислительные узлы), но не наоборот. Такая сеть называется сетью *прямого распространения* (feed-forward) или *ациклической* сетью (acyclic). На рис. 1.15 показана структура такой сети для случая четырех узлов в каждом из слоев (входном и выходном). Такая нейронная сеть называется *однослойной* (single-layer network), при этом под единственным слоем подразумевается слой вычислительных элементов (нейронов). При подсчете числа слоев мы не принимаем во внимание узлы источника, так как они не выполняют никаких вычислений.

Многослойные сети прямого распространения

Другой класс нейронных сетей прямого распространения характеризуется наличием одного или нескольких *скрытых слоев* (hidden layer), узлы которых называются *скрытыми нейронами* (hidden neuron), или *скрытыми элементами* (hidden unit). Функция последних заключается в посредничестве между внешним входным сигналом и выходом нейронной сети. Добавляя один или несколько скрытых слоев, мы можем выделить статистики высокого порядка. Такая сеть позволяет выделять *глобальные*

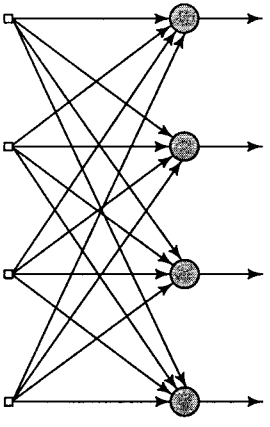


Рис. 1.15. Сеть прямого распространения с одним слоем нейронов

свойства данных с помощью локальных соединений за счет наличия дополнительных синаптических связей и повышения уровня взаимодействия нейронов [194]. Способность скрытых нейронов выделять статистические зависимости высокого порядка особенно существенна, когда размер входного слоя достаточно велик.

Узлы источника входного слоя сети формируют соответствующие элементы шаблона активации (входной вектор), которые составляют входной сигнал, поступающий на нейроны (вычислительные элементы) второго слоя (т.е. первого скрытого слоя). Выходные сигналы второго слоя используются в качестве входных для третьего слоя и т.д. Обычно нейроны каждого из слоев сети используют в качестве входных сигналов выходные сигналы нейронов только предыдущего слоя. Набор выходных сигналов нейронов выходного (последнего) слоя сети определяет общий отклик сети на данный входной образ, сформированный узлами источника входного (первого) слоя. Сеть, показанная на рис. 1.16, называется *сетью 10-4-2*, так как она имеет 10 входных, 4 скрытых и 2 выходных нейрона. В общем случае сеть прямого распространения с m входами, h_1 нейронами первого скрытого слоя, h_2 нейронами второго скрытого слоя и q нейронами выходного слоя называется *сетью* $m - h_1 - h_2 - q$.

Нейронная сеть, показанная на рис. 1.16, считается *полносвязной* (fully connected) в том смысле, что все узлы каждого конкретного слоя соединены со всеми узлами смежных слоев. Если некоторые из синаптических связей отсутствуют, такая сеть называется *неполносвязной* (partially connected).

Рекуррентные сети

Рекуррентная *нейронная сеть* (recurrent network) отличается от сети прямого распространения наличием по крайней мере одной *обратной связи* (feedback loop). Например, рекуррентная сеть может состоять из единственного слоя нейронов, каждый из которых направляет свой выходной сигнал на входы всех остальных нейронов слоя.

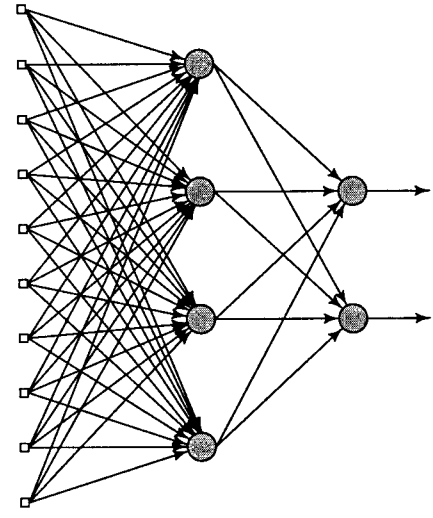


Рис. 1.16. Полносвязная сеть прямого распространения с одним скрытым и одним выходным слоем

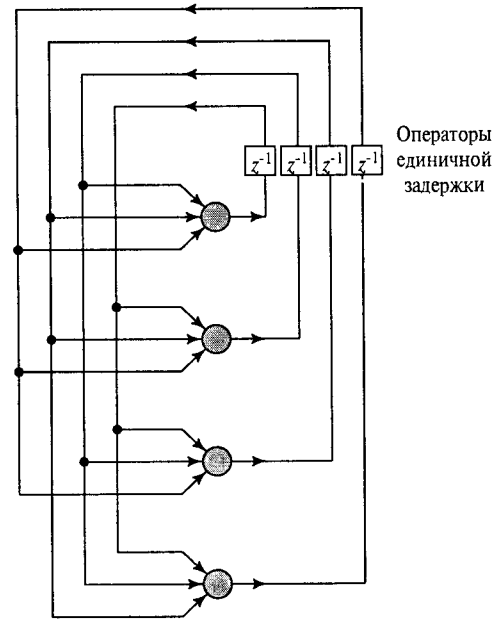


Рис. 1.17. Рекуррентная сеть без скрытых нейронов и обратных связей нейронов с самими собой

Архитектура такой нейронной сети показана на рис. 1.17. Обратите внимание, что в приведенной структуре отсутствуют обратные связи нейронов с самими собой. Рекуррентная сеть, показанная на рис. 1.17, не имеет скрытых нейронов. На рис. 1.18 показан другой класс рекуррентных сетей — со скрытыми нейронами. Здесь обратные связи исходят как из скрытых, так и из выходных нейронов.

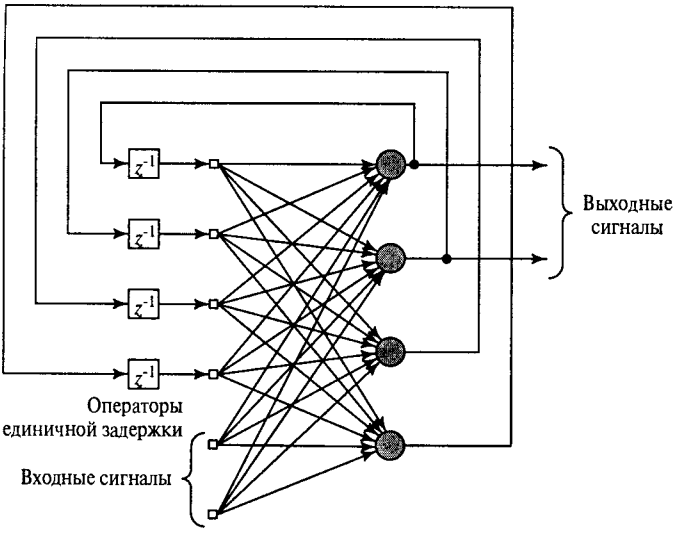


Рис. 1.18. Рекуррентная сеть со скрытыми нейронами

Наличие обратных связей в сетях, показанных на рис. 1.17 и 1.18, оказывает непосредственное влияние на способность таких сетей к обучению и на их производительность. Более того, обратная связь подразумевает использование *элементов единичной задержки* (unit-delay element) (они обозначены как z^{-1}), что приводит к нелинейному динамическому поведению, если, конечно, в сети содержатся нелинейные нейроны.

1.7. Представление знаний

В разделе 1.1 при определении нейронных сетей часто использовался термин "знания" без явного описания его значения. Этот пробел можно устранить, предложив следующее общее определение [295].

Под знаниями понимается хранимая информация или модели, используемые человеком или машиной для интерпретации, предсказания и реакции на внешние события.

К вопросам *представления знаний* (knowledge representation) относятся следующие: какую информацию необходимо хранить и как эту информацию представить физически для ее последующего использования. Таким образом, исходя из самой природы знаний, способ их представления определяется поставленной целью. Относительно реальных приложений "интеллектуальных" систем можно утверждать, что успех решения зависит от хорошего представления знаний [1166]. Это касается и нейронных сетей, представляющих собой отдельный класс интеллектуальных систем. Форма представления входных сигналов может быть самой разной. Это приводит к тому, что разработка приемлемых нейросетевых решений становится творческим процессом.

Основной задачей нейронной сети является наилучшее обучение модели окружающего мира для решения поставленной задачи. Знания о мире включают два типа информации.

1. Известное состояние окружающего мира, представленное имеющимися в наличии достоверными фактами. Такая информация называется *априорной* (prior).
2. Наблюдения за окружающим миром (измерения), полученные с помощью сенсоров, адаптированных для конкретных условий, в которых должна функционировать данная нейронная сеть. Обычно такие измерения в значительной мере зашумлены, что потенциально может стать источником ошибок. В любом случае измерения, полученные таким способом, формируют множество информации, *примеры* из которого используются для обучения нейронной сети.

Примеры могут быть *маркированными* (labeled) и *немаркированными* (unlabeled). В маркированных примерах *входному сигналу* (input signal) соответствует *желаемый отклик* (desired response). Немаркированные примеры состоят из нескольких различных реализаций одного входного сигнала. В любом случае набор примеров, будь то маркированных или нет, представляет собой знания об интересующей предметной области, на основании которых и проводится обучение нейронной сети.

Множество пар сигналов вход-выход, каждая из которых состоит из входного сигнала и соответствующего ему желаемого выхода, называют *обучающими данными* (training data) или *обучающей выборкой* (training sample). Для примера рассмотрим задачу *распознавания цифр* (digit recognition problem). В этой задаче входной сигнал (изображение) представляет собой матрицу, состоящую из черных и белых точек. Каждое изображение представляет одну из десяти рукописных цифр на белом фоне. Желаемым откликом сети является конкретная цифра, изображение которой подается в качестве входного сигнала. Обычно обучающая выборка состоит из большого числа рукописных цифр, что отражает ситуацию, которая может возникнуть в реальном мире. При наличии такого набора примеров нейронная сеть создается следующим образом.

- Во-первых, выбирается соответствующая нейросетевая архитектура, в которой размер входного слоя соответствует количеству пикселей на рисунке, а в выходном слое содержится десять нейронов, соответствующих цифрам. После этого выполняется настройка весовых коэффициентов сети на основе обучающего множества. Этот режим работы сети называется *обучением*.
- Во-вторых, эффективность обучения сети проверяется (тестируется) на множестве примеров, отличных от использованных при обучении. При этом на вход сети подается изображение, для которого известен целевой выход сети. Эффективность обучения сети проверяется путем сравнения результатов распознавания с реальными цифрами. Этот этап работы нейронной сети называют *обобщением* (generalization) (данный термин взят из психологии).

Здесь и кроется фундаментальное отличие между созданием нейронной сети и разработкой классических методов обработки информации для задач классификации. В последнем случае мы в первую очередь формулируем математическую модель исследуемой среды, верифицируем ее на реальных данных, а затем разрабатываем классификатор на основе этой модели. Создание нейронной сети основывается непосредственно на реальных данных, которые *говорят сами за себя*. Таким образом, нейронные сети не только реализуют полноценную модель среды, но и обеспечивают обработку данных.

Набор данных, используемый для обучения сети, должен содержать как *положительные*, так и *отрицательные* примеры. Например, в задаче пассивной эхо-локации положительные примеры включают сигналы, отраженные от интересующего объекта (например, подводной лодки). Однако в реальной среде на отклик радара влияют и морские объекты, случайно попавшие в зону сигнала. Чтобы понизить вероятность неверной трактовки сигнала, в множество примеров добавляют сигналы, полученные при отсутствии искомого объекта.

В нейронной сети заданной архитектуры знания об окружающей среде представляются множеством свободных параметров (т.е. синаптических весов и порогов) сети. Такая форма представления знаний соответствует самой природе нейронных сетей. Именно в ней кроется ключ эффективности нейросетевых моделей.

Вопрос представления знаний в нейронной сети является очень сложным. Тем не менее можно выделить четыре общих правила [48].

Правило 1. Сходные входные сигналы от схожих классов должны формировать единое представление в нейронной сети. Исходя из этого, они должны быть классифицированы как принадлежащие к одной категории.

Существует множество подходов к определению степени сходства входных сигналов. Обычно степень подобия определяется на основе *Евклидова расстояния* (Euclidian distance). Для примера предположим, что существует некоторый вектор $\mathbf{x}_i$ размерности m

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{im}]^T.$$

Все его элементы — действительные числа, а обозначение ^{T} указывает на то, что матрица *транспонирована* (transposition). Вектор $\mathbf{x}_i$ определяет некоторую точку в m-мерном *Евклидовом пространстве* (которое обозначается как $\Re^m$). *Евклидово расстояние* между парой m-мерных векторов $\mathbf{x}_i$ и $\mathbf{x}_j$ вычисляется как

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \left[\sum_{k=1}^{m} (x_{ik} - x_{jk})^2\right]^{1/2}, \qquad (1.23)$$

где $\mathbf{x}_{ik}$ и $\mathbf{x}_{jk}$ — k-е элементы векторов $\mathbf{x}_i$ и $\mathbf{x}_j$ соответственно. Отсюда следует, что степень сходства между входными сигналами, представленными векторами $\mathbf{x}_i$ и $\mathbf{x}_j$, является величиной, *обратной* Евклидову расстоянию между ними $d(\mathbf{x}_i, \mathbf{x}_j)$. Чем

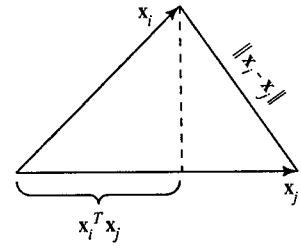


Рис. 1.19. Иллюстрация взаимосвязи между скалярным произведением и Евклидовым расстоянием, выбираемыми в качестве меры сходства образов

ближе друг к другу отдельные элементы векторов $\mathbf{x}_i$ и $\mathbf{x}_j$, тем меньше Евклидово расстояние $d(\mathbf{x}_i, \mathbf{x}_j)$ и тем выше сходство между векторами $\mathbf{x}_i$ и $\mathbf{x}_j$. Правило 1 констатирует, что если векторы $\mathbf{x}_i$ и $\mathbf{x}_j$ схожи, то они должны быть отнесены к одной категории (классу).

Еще один подход к определению степени сходства основывается на идее *скалярного произведения* (inner product), взятой из алгебры матриц. Если векторы $\mathbf{x}_i$ и $\mathbf{x}_j$ имеют одинаковую размерность, их скалярное произведение $\mathbf{x}_i^T\mathbf{x}_j$ определяется следующим образом:

$$(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j = \sum_{k=1}^{m} x_{ik}x_{jk}. \tag{1.24}$$

Результат деления скалярного произведения $(\mathbf{x}_i, \mathbf{x}_j)$ на $||\mathbf{x}_i||\ ||\mathbf{x}_j||$ равен косинусу внутреннего угла между векторами $\mathbf{x}_i$ и $\mathbf{x}_j$.

Эти два метода измерения сходства тесно связаны друг с другом (рис. 1.19). Евклидово расстояние $||\mathbf{x}_i - \mathbf{x}_j||$ между векторами $\mathbf{x}_i$ и $\mathbf{x}_j$ связано с проекцией вектора $\mathbf{x}_i$ на вектор $\mathbf{x}_j$. На рис. 1.19 видно, что чем меньше Евклидово расстояние $||\mathbf{x}_i - \mathbf{x}_j||$, тем больше скалярное произведение $\mathbf{x}_i^T\mathbf{x}_j$.

Чтобы формализовать это соотношение, нормализуем векторы $\mathbf{x}_i$ и $\mathbf{x}_j$. При этом их длина будет равна единице: $||\mathbf{x}_i|| = ||\mathbf{x}_j|| = 1$.

Используя выражение (1.23), запишем:

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j) = 2 - 2\mathbf{x}_i^T\mathbf{x}_j. \tag{1.25}$$

Из выражения (1.25) видно, что минимизация Евклидова расстояния $||\mathbf{x}_i - \mathbf{x}_j||$ и максимизация скалярного произведения $\mathbf{x}_i^T\mathbf{x}$ приводят к увеличению сходства между векторами $\mathbf{x}_i$ и $\mathbf{x}_j$.

Здесь Евклидово расстояние и скалярное произведение описаны в детерминистских терминах. А что будет, если векторы $\mathbf{x}_i$ и $\mathbf{x}_j$ взять из разных множеств данных? Для примера предположим, что различие между двумя множествами данных выражается в различии между векторами их математического ожидания. Пусть векторы $\boldsymbol{\mu}_i$ и $\boldsymbol{\mu}_j$ определяют средние значения векторов $\mathbf{x}_i$ и $\mathbf{x}_j$ соответственно, т.е.

$$\boldsymbol{\mu}_i = E[\mathbf{x}_i], \tag{1.26}$$

где E — статистический оператор математического ожидания. Вектор $\boldsymbol{\mu}_j$ определяется аналогичным способом. Для измерения расстояния между двумя множествами можно использовать *расстояние Махаланобиса* (Mahalanobis distance), которое обозначается как d_{ij}. Квадрат этой величины определяется следующей формулой [269]:

$$d_{ij}^2 = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \Sigma^{-1} (\mathbf{x}_j - \boldsymbol{\mu}_j), \tag{1.27}$$

где Σ^{-1} — обратная матрица для матрицы ковариации $\boldsymbol{\sigma}$. Предполагается, что матрица ковариации для обоих множеств одна и та же, т.е.

$$\Sigma = E[(\mathbf{x}_i - \boldsymbol{\mu}_i)(\mathbf{x}_i - \boldsymbol{\mu}_i)^T] = E[(\mathbf{x}_j - \boldsymbol{\mu}_j)(\mathbf{x}_j - \boldsymbol{\mu}_j^T)]. \tag{1.28}$$

В частном случае, когда $\mathbf{x}_i = \mathbf{x}_j$, $\boldsymbol{\mu}_i = \boldsymbol{\mu}_j = \boldsymbol{\mu}$ и $\boldsymbol{\Sigma} = \mathbf{I}$, где $\mathbf{I}$ — единичная матрица, расстояние Махаланобиса вырождается в Евклидово расстояние между вектором $\mathbf{x}_i$ и вектором математического ожидания $\boldsymbol{\mu}$.

Правило 2. Элементы, отнесенные к различным классам, должны иметь в сети как можно более отличные представления.

Это правило прямо противоположно первому.

Правило 3. Если некоторое свойство имеет важное значение, то для его представления в сети необходимо использовать большое количество нейронов.

Для примера рассмотрим задачу обнаружения радаром некоторого объекта (например, самолета) при наличии помех (вызванных отражением сигнала от посторонних объектов, таких как дома, деревья, облака и т.п.). Эффективность такой радарной системы измеряется двумя вероятностными величинами.

- *Вероятность обнаружения* (probability of detection) — вероятность того, что при наличии объекта система его обнаружит.
- *Вероятность ложной тревоги* (probability of false alarm) — вероятность того, что система определит наличие объекта, когда того на самом деле не существует.

В соответствии с *критерием Неймана–Пирсона* (Neyman-Pearson criterion) вероятность обнаружения должна быть максимальной, а вероятность ложной тревоги не должна превосходить некоторой заданной величины [1080]. В подобных приложениях очень важно, чтобы во входном сигнале содержалась информация о цели. Правило 3 констатирует, что в процесс принятия решения о наличии цели в полученном сигнале должно быть вовлечено большое число нейронов. В любом случае увеличение количества нейронов повышает достоверность принятого решения и устойчивость к помехам.

Правило 4. В структуру нейронной сети должны быть встроены априорная информация и инварианты, что упрощает архитектуру сети и процесс ее обучения.

Это правило играет особую роль, поскольку правильная конфигурация сети обеспечивает ее *специализацию* (specialized structure), что очень важно по следующим причинам [917].

1. Биологические сети, обеспечивающие обработку зрительной и слуховой информации, сильно специализированы.
2. Нейронная сеть со специализированной структурой обычно включает значительно меньшее количество свободных параметров, которые нужно настраивать, чем полносвязная сеть. Из этого следует, что для обучения специализированной сети требуется меньше данных. При этом на обучение затрачивается меньше времени, и такая сеть обладает лучшей обобщающей способностью.
3. Специализированные сети обладают большей пропускной способностью.
4. Стоимость создания специализированных нейронных сетей сокращается, поскольку их размер существенно меньше размера полносвязных сетей.

Как встроить априорную информацию в структуру нейронной сети

Для выполнения правила 4 необходимо понять, как разработать специализированную структуру, в которую встроена априорная информация. К сожалению, в настоящее время не существует четкого решения этой задачи. Однако можно предложить некоторые специальные процедуры, которые, как показывает практика, приводят к хорошим результатам. В частности, можно использовать комбинацию двух следующих приемов [621].

1. *Ограничение сетевой архитектуры* (restricting network architecture) с помощью локальных связей, называемых *рецепторными полями* (receptive fields)[5].
2. *Ограничение выбора синаптических весов* за счет *совместного использования весов* (weight-sharing)[6].

Эти два приема, особенно последний, обеспечивают одно важное преимущество — значительно сокращают количество свободных параметров сети.

Для примера рассмотрим неполносвязную сеть прямого распространения, показанную на рис. 1.20. Эта сеть имеет ограниченную архитектуру. Первые шесть узлов источника образуют рецепторное поле скрытого нейрона с номером 1, и т.д. для всех

---

[5] Согласно утверждению Кафлера [604], термин "рецепторное поле" изначально был введен в [978], а позднее повторно использован в [421]. В контексте систем обработки зрительной информации под рецепторным полем нейрона понимается ограниченная область сетчатки, которая выполняет функцию передачи этому нейрону сигналов от светочувствительных элементов.

[6] Принцип совместного использования весов был впервые описан Руммельхартом в [915].

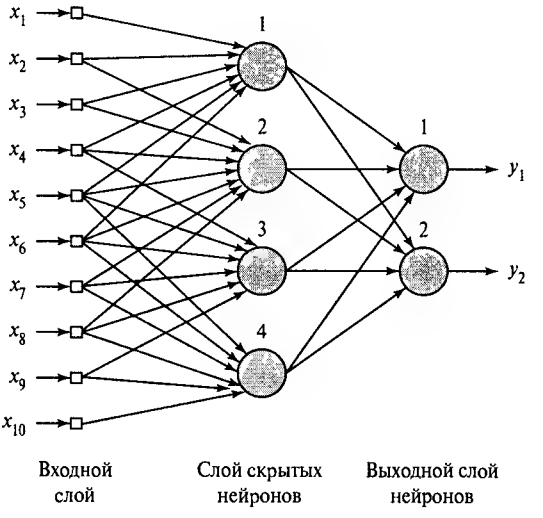


Рис. 1.20. Пример сети с рецепторными полями и совместным использованием весов. Все четыре скрытых нейрона для реализации синаптических связей совместно используют одно и то же множество весов

остальных скрытых нейронов сети. Совместное использование весов состоит в применении одинакового набора синаптических весов для каждого из нейронов скрытого слоя сети. Учитывая, что каждый из четырех скрытых нейронов имеет по шесть локальных связей (см. рис. 1.20), индуцированное локальное поле скрытого нейрона j можно описать следующим выражением:

$$v_j = \sum_{i=1}^{6} w_i x_{i+j-1}, \quad j = 1,\ 2,\ 3,\ 4, \tag{1.29}$$

где $\{w_i\}_{i=1}^{6}$ определяет один и тот же набор весов, совместно используемых всеми четырьмя скрытыми нейронами, а x_k является сигналом, получаемым из узла источника с номером $k = i + j - 1$. Выражение (1.29) записано в форме *суммы свертки* (convolution sum). Именно по этой причине сеть прямого распространения с локальными связями и совместно используемыми весами называют *сетью свертки* (convolution network).

Первая часть правила 4 определяет необходимость встраивания априорной информации в структуру нейронной сети, а вторая часть этого правила касается вопроса инвариантов.

Как встроить инварианты в структуру нейронной сети

Рассмотрим следующие физические явления.

- Если исследуемый объект вращается, то соответствующим образом меняется и его образ, воспринимаемый наблюдателем.
- В когерентном радаре, обеспечивающем информацию об амплитуде и фазе источников окружающей среды, эхо от движущегося объекта смещено по частоте. Это связано с эффектом Доплера, который возникает при радиальном движении объекта наблюдения относительно радара.
- Диктор может произносить слова как тихим, так и громким голосом, как медленно, так и скороговоркой.

Для того чтобы создать систему распознавания объекта, речи или эхо-локации, учитывающую явления такого рода, необходимо принимать во внимание диапазон *трансформаций* (transformation) наблюдаемого сигнала [94]. Соответственно основным требованием при распознавании образов является создание такого классификатора, который *инвариантен* к этим трансформациям. Другими словами, на результат классификации не должны оказывать влияния трансформации входного сигнала, поступающего от объекта наблюдения.

Существуют как минимум три приема обеспечения инвариантности нейронной сети классификации к подобным трансформациям [94].

1. *Структурная инвариантность* (invariance by structure). Инвариантность может быть привнесена в нейронную сеть с помощью соответствующей структуризации. В частности, синаптические связи между отдельными нейронами сети строятся таким образом, чтобы трансформированные версии одного и того же сигнала вызывали один и тот же выходной сигнал. Рассмотрим для примера нейросетевую классификацию входного сигнала, которая должна быть инвариантна по отношению к плоскому вращению изображения относительно своего центра. Структурную инвариантность сети относительно вращения можно выразить следующим образом. Пусть w_{ji} — синаптический вес нейрона j, связанного с пикселем i входного изображения. Если условие $w_{ji} = w_{jk}$ выполняется для всех пикселей j и k, лежащих на равном удалении от центра изображения, нейронная сеть будет инвариантной к вращению. Однако, для того чтобы обеспечить инвариантность относительно вращения, нужно дублировать синаптические веса w_{ji} всех пикселей, равноудаленных от центра изображения. Недостатком структурной инвариантности является то, что количество синаптических связей изображения даже среднего размера будет чрезвычайно велико.

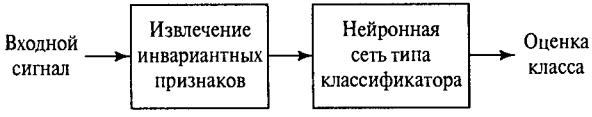
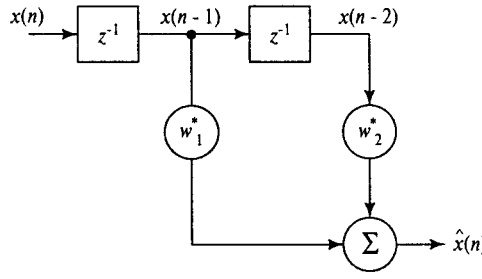


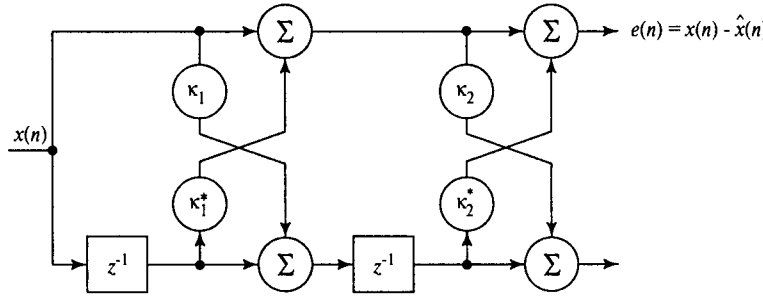
Рис. 1.21. Диаграмма системы, использующей пространство инвариантных признаков

2. *Инвариантность по обучению* (invariance by training). Нейронные сети обладают естественной способностью классификации образов. Эту способность можно использовать для обеспечения инвариантности сети к трансформациям. Сеть обучается на множестве примеров одного и того же объекта, при этом в каждом примере объект подается в несколько измененном виде (например, снимки с разных ракурсов). Если количество таких примеров достаточно велико и если нейронная сеть обучена отличать разные точки зрения на объект, можно ожидать, что эти данные будут обобщены и сеть сможет распознавать ракурсы объекта, которые не использовались при обучении. Однако с технической точки зрения инвариантность по обучению имеет два существенных недостатка. Во-первых, если нейронная сеть была научена распознавать трансформации объектов некоторого класса, совсем не обязательно, что она будет обладать инвариантностью по отношению к трансформациям объектов других классов. Во-вторых, такое обучение является очень ресурсоемким, особенно при большой размерности пространства признаков.
3. *Использование инвариантных признаков* (invariant feature space). Третий метод создания инвариантного нейросетевого классификатора проиллюстрирован на рис. 1.21. Он основывается на предположении, что из входного сигнала можно выделить информативные признаки, которые описывают самую существенную информацию, содержащуюся в наборе данных, и при этом инвариантны к трансформациям входного сигнала. При использовании таких признаков в нейронной сети не нужно хранить лишний объем информации, описывающей трансформации объекта. В самом деле, при использовании инвариантных признаков отличия между разными экземплярами одного и того же объекта могут быть вызваны только случайными факторами, такими как шум. Использование пространства инвариантных признаков имеет три важных преимущества. Во-первых, уменьшается количество признаков, которые подаются в нейронную сеть. Во-вторых, ослабляются требования к структуре сети. И, в-третьих, гарантируется инвариантность всех объектов по отношению к известным трансформациям [94]. Однако этот подход требует хорошего знания специфики проблемы.

Итак, из вышесказанного можно сделать вывод, что использование инвариантных признаков является наиболее подходящим методом для обеспечения инвариантности нейросетевых классификаторов.



а)



б)

Рис. 1.22. Модель авторегрессии второго порядка: модель фильтра на линии задержки с отводами (а) и модель решетчатого фильтра (б). (Звездочкой отмечено комплексное сопряжение.)

Чтобы проиллюстрировать идею пространства инвариантных признаков, рассмотрим в качестве примера систему когерентного радара, используемую авиадиспетчерами, во входном сигнале которой может содержаться информация, поступающая от самолетов, стаи птиц и некоторых погодных явлений. Сигнал радара, отраженный от различных целей, имеет разные спектральные характеристики. Более того, экспериментальные исследования показали, что сигнал такого радара можно промоделировать с помощью *авторегрессионного процесса* (AR-процесса) *среднего порядка* (autoregressive process of moderate order) [439]. AR-процесс представляет собой особый вид регрессионной модели, описываемой следующим образом:

$$x(n) = \sum_{i=1}^{M} a_i^* x(n-i) + e(n), \tag{1.30}$$

где $\{a_i\}_{i=1}^{M}$ — *коэффициенты* (coefficient) авторегрессии; M — *порядок модели* (model order); $x(n)$ — *входной сигнал* (input signal); $e(n)$ — *помеха* (error), представляющая собой белый шум. Модель, описанная формулой (1.30), представляет собой *фильтр на линии задержки с отводами* (tapped-delay-line filter), показанный на рис. 1.22, *а* для $M = 2$. Аналогично, ее можно представить как *решетчатый фильтр* (lattice filter), показанный на рис. 1.22, *б*, коэффициенты которого называются *коэффици-*

ентами отражения (reflection coefficient). Между коэффициентами авторегрессии (рис. 1.22, *а*) и коэффициентами отражения (рис. 1.22, *б*) существует однозначное соответствие. В обеих моделях предполагается, что входной сигнал $x(n)$ является комплексной величиной (как в случае с когерентным радаром), в которой коэффициенты авторегрессии и коэффициенты отражения также являются комплексными. Звездочка в выражении (1.30) и на рис. 1.22 обозначает *комплексное сопряжение*. Здесь важно подчеркнуть, что данные когерентного радара можно описать множеством коэффициентов авторегрессии или соответствующим ему множеством коэффициентов отражения. Последнее имеет определенные преимущества в плане сложности вычислений. Для него существуют эффективные алгоритмы получения результата непосредственно из входных данных. Задача выделения признаков усложняется тем фактом, что движущиеся объекты характеризуются переменными доплеровскими частотами, которые зависят от скорости объекта относительно радара и создают искажения в спектре коэффициентов отражения, по которым определяются признаки. Для того чтобы обойти эту сложность, в процессе вычисления коэффициентов отражения следует использовать *инвариантность Доплера* (Doppler invariance). Угол фазы первого коэффициента отражения принимается равным доплеровской частоте сигнала радара. Соответственно для всех коэффициентов выполняется *нормализация* относительно доплеровской частоты, устраняющая влияние сдвига доплеровской частоты. Для этого определяется новое множество коэффициентов отражения κ_m, связанных с множеством исходных коэффициентов отражения κ_m следующим соотношением:

$$\kappa'_m = \kappa_m e^{-jm\theta}, \quad m = 1,\ 2,\ \ldots,\ M, \tag{1.31}$$

где θ — фазовый угол первого коэффициента отражения. Операция, описанная выражением (1.31), называется *гетеродинированием* (heterodyning). Исходя из этого, набор *инвариантных к смещению Доплера признаков* (Doppler-invariant radar feature) представляется нормализованными коэффициентами отражения $\kappa'_1, \kappa'_2, \ldots, \kappa'_m$, где κ'_1 — единственный коэффициент этого множества с вещественным значением. Как уже отмечалось, основными категориями объектов, выделяемых радарной установкой, являются стаи птиц, самолеты, погодные явления и поверхность земли. Первые три категории объектов являются движущимися, в то время как последняя — нет. Гетеродинные спектральные параметры эха радара от земли аналогичны соответствующим параметрам эха от самолета. Отличить эти два сигнала можно по наличию у эха от самолета небольшого смещения Доплера. Следовательно, классификатор радара должен содержать постпроцессор (рис. 1.23). Он обрабатывает результаты классификации с целью идентификации класса земли [439]. *Препроцессор* (preprocessor), показанный на рис. 1.23, обеспечивает инвариантность признаков по отношению к смещению Доплера, в то время как *постпроцессор* использует смещение Доплера для разделения объектов "самолет" и "земля" в выходном сигнале.

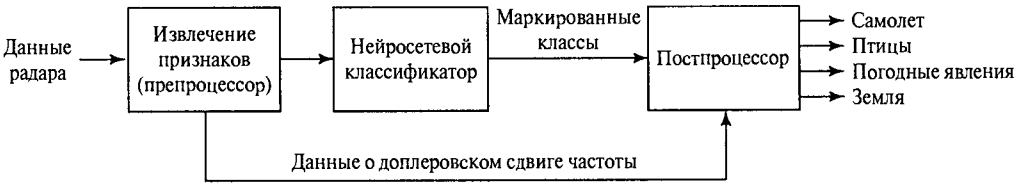


Рис. 1.23. Инвариантный к смещению Доплера классификатор сигнала радара

Гораздо более примечательным примером представления знаний в нейронной сети является биологическая система эхо-локации летучей мыши. Многие мыши используют сигналы с *частотной модуляцией* (frequency modulation), или *FM-сигналы* (frequency modulated signal), для создания акустической картины окружающего пространства. Несущая частота этого сигнала изменяется во времени. В частности, летучая мышь ртом испускает короткие FM-сигналы, а органы слуха использует в качестве приемника эха. Эхо от интересующей цели представляется в слуховом аппарате активностью нейронов, отвечающих за различные наборы акустических параметров. В слуховом аппарате летучей мыши информация представляется тремя основными характеристиками [993], [994].

- *Частота эхо-сигнала* (echo frequency). Кодируется частотной картой ушной улитки. Она сохраняется на протяжении всего пути сигнала по слуховому аппарату и выделяется отдельными нейронами, настроенными на определенные частоты.
- *Амплитуда эхо-сигнала* (echo amplitude). Кодируется другими нейронами, имеющими разные динамические характеристики. Они выделяют амплитудную характеристику и количество отзывов, пришедших в ответ на один сигнал запроса.
- *Задержка эхо-сигнала* (echo delay). Кодируется с помощью нейронных вычислений (основанных за взаимной корреляции).

Двумя основными характеристиками, используемыми для формирования изображения, являются *спектр* (spectrum) для данной формы объекта и *задержка* (delay). Летучая мышь формирует "форму" объекта в терминах времени получения отраженного сигнала от различных отражающих поверхностей объекта. Для этого частотная информация, содержащаяся в спектре эхо-сигнала, преобразуется в оценки *временной структуры* (time structure) объекта. Эксперименты, проведенные Симмонсом (Simmons) и его коллегами с *большой бурой летучей мышью* Eptesicus fuscus, показали, что этот процесс состоит из временных и частотно-временных преобразований, в результате которых формируется общая задержка для воспринимаемого объекта. Таким образом, частотная и временная информация, получаемая органами чувств летучей мыши за счет последовательности преобразований, приводится к единому виду. Более того, в процессе формирования образа используются инвариантные признаки, что делает его независимым от перемещения объекта и движения самой летучей мыши.

Возвращаясь к главной теме этого раздела (представление знаний в нейронной сети), можно сказать, что этот вопрос непосредственно связан с сетевой архитектурой, описанной в разделе 1.6. К сожалению, в настоящее время не существует какой-либо формализованной теории оптимизации структуры нейронных сетей или оценки влияния архитектуры сети на представление знаний в ней. Ответы на эти вопросы обычно получают экспериментальным путем. При этом сам разработчик нейронной сети становится важным элементом цикла структурного обучения.

Независимо от того, как выбирается архитектура сети, знания о предметной области выделяются нейронной сетью в процессе обучения. Эти знания представляются в компактно распределенном виде весов синаптических связей сети. Такая форма представления знаний позволяет нейронной сети адаптироваться и выполнять обобщение, однако не обеспечивает полноценного описания вычислительного процесса, используемого для принятия решения или формирования выходного сигнала. Это может накладывать серьезные ограничения на использование нейросетевого подхода, особенно в тех областях, где решающим является принцип безопасности, например в области диспетчеризации движения самолетов или в медицинской диагностике. В таких приложениях не только желательно, но и жизненно необходимо обеспечить *возможность объяснения* (explanation capability). Одним из способов обеспечения такой возможности является интеграция нейронных сетей и моделей искусственного интеллекта в единую гибридную систему. Этот вопрос будет обсуждаться в следующем разделе.

1.8. Искусственный интеллект и нейронные сети

Основной задачей *искусственного интеллекта* (artificial intelligence — AI) является разработка парадигм или алгоритмов, обеспечивающих компьютерное решение когнитивных задач, свойственных человеческому мозгу [924]. Следует заметить, что это определение искусственного интеллекта не является единственно возможным.

Системы искусственного интеллекта должны обеспечивать решение следующих трех задач: накопление знаний, применение накопленных знаний для решения проблемы и извлечение знаний из опыта. Системы искусственного интеллекта реализуют три ключевые функции: представление, рассуждение и обучение [924] (рис. 1.24).

1. *Представление* (representation). Одной из отличительных черт систем искусственного интеллекта является использование *символьного* языка (symbol structure) для представления общих знаний о предметной области и конкретных знаний о способах решения задачи. Символы обычно формулируются в уже известных терминах. Это делает символьное представление относительно простым и понятным человеку. Более того, понятность символьных систем искусственного интеллекта делает их пригодными для человеко-машинного общения.

Рис. 1.24. Три ключевые функции систем искусственного интеллекта

Термин "знания", используемый создателями систем искусственного интеллекта, является всего лишь еще одним названием данных. Знания могут иметь процедурный и декларативный характер. В *декларативном* (declarative) представлении знания — это статический набор фактов. При этом существует относительно малый объем процедур, используемых для манипуляций этими фактами. Характерной особенностью декларативного представления является то, что в глазах человека оно имеет смысл само по себе, независимо от использования в системах искусственного интеллекта. В *процедурном* (procedural) представлении знания внедрены в процедуры, функционирующие независимо от смысла самих знаний. В большинстве предметных областей требуются одновременно оба типа представления знаний.

2. *Рассуждения* (reasoning). Под рассуждениями обычно понимается способность решать задачи. Для того чтобы систему можно было назвать разумной, она должна удовлетворять следующим условиям [295].

- Описывать и решать широкий спектр задач.
- Понимать *явную* (explicit) и *неявную* (implicit) информацию.
- Иметь механизм *управления* (control), определяющий операции, выполняемые для решения отдельных задач.

Решение задач можно рассматривать как некоторую задачу *поиска* (searching problem). В процессе поиска используются *правила* (rules), *данные* (data) и *управляющие воздействия* (control) [785]. Правила действуют на области данных, а управляющие воздействия определяются для правил. Для примера рассмотрим известную "задачу коммивояжера". В ней требуется найти кратчайший маршрут из одного города в другой. При этом все города, расположенные по маршруту, необходимо посетить только один раз. В этой задаче множество данных состоит из всех возможных маршрутов и их стоимостей, представленных в форме взвешенного графа. Правила определяют пути движения из одного города в другой, а модуль управления решает, когда и какие правила применять.

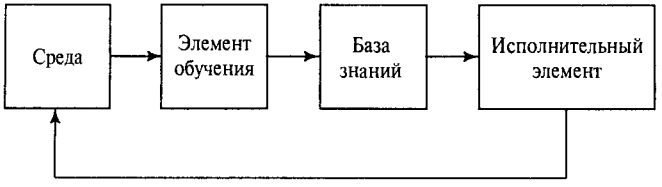


Рис. 1.25. Простейшая модель машинного обучения

Во многих практических задачах (например, в медицинской диагностике) доступный набор знаний является неполным или неточным. В таких ситуациях используются *вероятностные рассуждения* (probabilistic reasoning), позволяющие системам искусственного интеллекта работать в условиях неопределенности [822], [916].

3. *Обучение* (learning). В простейшей модели машинного обучения (рис. 1.25) информацию для *обучаемого элемента* (learning element) предоставляет сама среда. Обучаемый элемент использует полученную информацию для модернизации *базы знаний* (knowledge base), знания из которой *функциональный элемент* (performance element) затем использует для выполнения поставленной задачи. Информация, поступающая из внешней среды, является несовершенной, поэтому обучаемый элемент заранее не знает, как заполнить пробелы или игнорировать несущественные детали. Машина действует наугад, после чего получает сигнал *обратной связи* (feedback) от функционального элемента. Механизм обратной связи позволяет системе проверять рабочие гипотезы и пересматривать их по мере необходимости.

Машинное обучение может включать два совершенно разных способа обработки информации: *индуктивный* (inductive) и *дедуктивный* (deductive). При индуктивной обработке информации общие шаблоны и правила создаются на основании практического опыта и потоков данных. При дедуктивной обработке информации для определения конкретных фактов используются общие правила. Обучение на основе подобия представляет собой индуктивный процесс, а доказательство теорем — дедуктивный, поскольку оно опирается на известные аксиомы и уже доказанные теоремы. В обучении на основе объяснения используется как индукция, так и дедукция.

Возникающие при обучении сложности и накопленный при этом опыт привели к созданию различных методов и алгоритмов пополнения баз знаний. В частности, если в данной предметной области работают опытные профессионалы, проще получить их обобщенный опыт, чем пытаться дублировать экспериментальный путь, который они прошли в процессе его накопления. Эта идея и положена в основу *экспертных систем* (expert system).

Возникает вопрос: как сравнить когнитивные модели нейронных сетей с символьными системами искусственного интеллекта? Для такого сравнения разобьем проблему на три части: уровень объяснения, стиль обработки и структуру представления [724].

1. *Уровень объяснения* (explanation level). Классические системы искусственного интеллекта (artificial intelligence — AI) основаны на символьном представлении. С точки зрения познания AI предполагает существование ментального представления, в котором познание осуществляется как *последовательная обработка* (sequential processing) символьной информации [780].

   В центре внимания нейронных сетей находятся модели *параллельной распределенной обработки* (parallel distributed processing или PDP). В этих моделях предполагается, что обработка информации происходит за счет взаимодействия большого количества нейронов, каждый из которых передает сигналы возбуждения и торможения другим нейронам сети [912]. Более того, в теории нейронных сетей большое внимание уделяется нейробиологическому описанию процесса познания.

2. *Стиль обработки* (processing style). В классических системах искусственного интеллекта обработка происходит *последовательно* (sequential), как и в традиционном программировании. Даже если порядок выполнения действий строго не определен (например, при сканировании правил и фактов в экспертных системах), операции все равно выполняются пошагово. Такая последовательная обработка, скорее всего, объясняется последовательной природой естественных языков и логических заключений, а также структурой машины фон Неймана. Нельзя забывать о том, что классические системы искусственного интеллекта зародились практически в ту же интеллектуальную эру, что и машина фон Неймана.

   В отличие от них, концепция обработки информации в нейронных сетях проистекает из принципа *параллелизма* (parallelism), который является источником их гибкости. Более того, параллелизм может быть массовым (сотни тысяч нейронов), что придает нейронным сетям особую форму робастности. Если вычисления распределены между множеством нейронов, практически не важно, что состояние отдельных нейронов сети отличается от ожидаемого. Зашумленный или неполный входной сигнал все равно можно распознать; поврежденная сеть может продолжать выполнять свои функции на удовлетворительном уровне, а обучение не обязательно должно быть совершенным. Производительность сети в пределах некоторого диапазона снижается достаточно медленно. Кроме того, можно дополнительно повысить робастность сети, представляя каждое свойство группой нейронов [459].

3. *Структура представления* (representational structure). В классических системах искусственного интеллекта в качестве модели выступает язык мышления, поэтому символьное представление имеет квази-лингвистическую структуру. Подобно фразам обычного языка, выражения классических систем искусственного интеллекта, как правило, сложны и составляются путем систематизации простых символов. Учитывая ограниченное количество символов, новые смысловые выражения строятся на основе композиции символьных выражений и аналогии между синтаксической структурой и семантикой.

С другой стороны, в нейронных сетях природа и структура представления являются ключевыми проблемами. В марте 1988 года в специальном выпуске журнала *Cognition* [300] были опубликованы критические замечания по поводу вычислительной адекватности нейронных сетей при решении когнитивных и лингвистических задач. Они аргументированы тем, что нейронные сети не удовлетворяют двум основным критериям процесса познания: природе *мысленного представления* (mental representation) и *мыслительных процессов* (mental process). В соответствии с этой работой следующие свойства присущи именно системам искусственного интеллекта и не присущи нейронным сетям.

- Мысленное представление характеризуется комбинаторной избирательной структурой и комбинаторной семантикой.
- Мыслительные процессы характеризуются чувствительностью к комбинаторной структуре представления, с которым они работают.

Таким образом, символьные модели искусственного интеллекта — это формальные системы, основанные на использовании языка алгоритмов и представлении данных по принципу "сверху вниз" (top-down), а нейронные сети — это параллельные распределенные процессоры, обладающие естественной способностью к обучению и работающие по принципу "снизу вверх" (bottom-up). Поэтому при решении когнитивных задач целесообразно создавать *структурированные модели на основе связей* (structured connectionist models) или *гибридные системы* (hybrid system), объединяющие оба подхода. Это обеспечит сочетание свойств адаптивности, робастности и единообразия, присущих нейронным сетям, с представлениями, умозаключениями и универсальностью систем искусственного интеллекта [293], [1108]. Для реализации этого подхода были разработаны методы извлечения правил из обученных нейронных сетей [61]. Эти результаты не только позволяют интегрировать нейронные сети с интеллектуальными машинами, но и обеспечивают решение следующих задач.

- Верификация нейросетевых компонентов в программных системах. Для этого внутреннее состояние нейронной сети переводится в форму, понятную пользователям.
- Улучшение обобщающей способности нейронной сети за счет выявления областей входного пространства, не достаточно полно представленных в обучающем множестве, а также определения условий, при которых обобщение невозможно.
- Выявление скрытых зависимостей на множестве входных данных.
- Интеграция символьного и коннекционистского подходов при разработке интеллектуальных машин.
- Обеспечение безопасности систем, для которых она является критичной.

1.9. Историческая справка

Эта вводная глава не может обойтись без небольшой исторической справки[7].

Современная эра нейронных сетей началась с новаторской работы Мак-Каллока и Питца [714]. Мак-Каллок по образованию был психиатром и нейроанатомом и более 20 лет занимался вопросами представления событий в нервной системе. Питц был талантливым математиком; он присоединился к исследованиям Мак-Каллока в 1942 году. Согласно [868], в статье Мак-Каллока и Питца описаны результаты, полученные в течение 5 лет группой специалистов по нейромоделированию из университета Чикаго, возглавляемой Рашевским (Rashevcky).

В своей классической работе Мак-Каллок и Питц описали логику вычислений в нейронных сетях на основе результатов нейрофизиологии и математической логики. Формализованная модель нейрона соответствовала принципу "все или ничего". Ученые показали, что сеть, составленная из большого количества таких элементарных процессорных единиц, соединенных правильно сконфигурированными и синхронно работающими синаптическими связями, принципиально способна выполнять любые вычисления. Этот результат был реальным прорывом в области моделирования нервной системы. Именно он явился причиной зарождения таких направлений в науке, как искусственный интеллект и нейронные сети.

В свое время работа Мак-Каллока и Питца широко обсуждалась. Она остается актуальной и сегодня. Эта работа оказала влияние на идеи фон Неймана, воплощенные в конструкции компьютера EDVAC (Electronic Discrete Variable Automatic Computer), разработанного на основе устройства ENIAC (Electronic Numerical Integrator and Computer) [74]. ENIAC был первым компьютером общего назначения. Он создавался с 1943 по 1946 год в университете штата Пенсильвания. Фон Нейман постоянно излагал теорию формальных нейронных сетей Мак-Каллока–Питца во второй из своих четырех лекций, которые он читал в Университете штата Иллинойс в 1949 году.

В 1948 году была издана знаменитая книга Винера (Weiner) под названием *Кибернетика*. В ней были описаны некоторые важные концепции управления, коммуникаций и статистической обработки сигналов. Вторая редакция этой книги вышла в 1961 году. В нее был добавлен материал, касающийся обучения и самоорганизации систем. Во второй главе обоих изданий этой книги Винер подчеркнул физическую значимость статистических механизмов в контексте излагаемой проблемы. Однако только через 30 лет в работах Хопфилда (Hopfield) был построен мост между статистическими механизмами и обучаемыми системами.

---

[7] Историческая справка в основном (но не полностью) основывается на [48], [53], [68], [74], [227], [390], [868], [916], [919], [964], [1042], [1142].

Следующей важной вехой в развитии нейронных сетей стал выход в свет в 1949 году книги Хебба (Hebb) *The Organization of Behavior* (Организация поведения). В ней приводится четкое определение физиологического правила обучения для *синаптической модификации* (synaptic modification). В частности, Хебб предположил, что, по мере того как организм обучается различным функциональным задачам, связи в мозге постоянно изменяются и при этом формируются *ансамбли нейронов* (neuron assembly). Знаменитый *постулат обучения* (postulate of learning) Хебба гласит, что эффективность переменного синапса между двумя нейронами повышается при многократной активации этих нейронов через данный синапс. Эта книга оказала влияние на развитие психологии, но, к сожалению, практически не возымела влияния на сообщество инженеров.

Книга Хебба стала источником вдохновения при создании вычислительных моделей *обучаемых и адаптивных систем* (learning and adaptive system). Первой попыткой использования компьютерного моделирования для проверки формализованной теории нейронов, основанной на постулате обучения Хебба, была, пожалуй, [894]. Результаты моделирования четко показали, что для полноты этой теории к ней следует добавить торможение. В том же году Аттли (Uttley) [1069] продемонстрировал, что нейронные сети с изменяемыми синапсами можно обучить классификации простейших двоичных образов (растровых изображений). Он ввел понятие и *активации нейрона*, которое было формально проанализировано Кайанелло (Caianielo) в 1961 году [167]. В своей более поздней работе (1979) Аттли высказал гипотезу о том, что эффективность переменного синапса в нервной системе зависит от статистических связей между переменными состояниями на другом конце синапса, связав, таким образом, теорию нейронных сетей с теорией информации Шеннона (Shannon).

В 1953 году вышла в свет книга, которая не потеряла своей актуальности и сегодня [53]. Идея, представленная в этой работе, состояла в том, что адаптивное поведение является не врожденным, а приобретенным, и с помощью обучения можно улучшить поведение животного (системы). Эта книга фокусировала внимание исследователей на динамических аспектах живого организма как системы и на понятии устойчивости.

В 1954 году Минский (Minsky) написал докторскую диссертацию в Принстонском университете, посвященную теории нейроаналоговых систем обучения с подкреплением и ее применению в задачах моделирования мозга [743]. В 1961 году была опубликована его работа, посвященная искусственному интеллекту [742]. Она называлась *Steps Toward Artificial Intelligence* (На пути к искусственному интеллекту). Эта работа содержала большой раздел, посвященный области, которая сейчас называется теорией нейронных сетей. В своей работе *Computation: Finite and Infinite Machines* (Вычисления: конечные и бесконечные машины) [741] Минский расширил результаты, полученные в 1943 году Мак-Каллоком и Питцом, и поместил их в контекст теории автоматов и теории вычислений.

В том же 1954 году Габор (Gabor), один из пионеров в теории коммуникации и первооткрыватель голографии, предложил идею *нелинейного адаптивного фильтра* (nonlinear adaptive filter) [330]. Со своими единомышленниками он создал машину, которая обучалась на примере стохастического процесса [331].

В 1950-х годах Тейлор (Tailor) инициировал работы по исследованию *ассоциативной памяти* (associative memory) [1044], а в 1961 году Стейнбах (Steinbuch) разработал *матрицу обучения* (learning matrix), состоящую из плоской сети переключателей, объединявшей массивы сенсорных рецепторов и моторных исполнительных механизмов [1015]. В 1969 году была опубликована хорошая работа по неголографической ассоциативной памяти [1158]. В ней представлены две модели: простая оптическая модель корреляционной памяти и нейросетевая модель, реализованная в виде оптической памяти. Среди других работ, которые внесли заметный вклад в раннее развитие ассоциативной памяти, следует отметить [50], [580], [771], в которых независимо друг от друга в одном и том же году описана идея *памяти на основе матрицы корреляции* (correlation matrix memory), которая строится на обучении по правилу внешнего произведения (outer product learning).

Одной из самых известных фигур в науке первой половины XX века был фон Нейман. *Архитектура фон Неймана* (Von Neumann architecture) является основой для создания цифровых компьютеров, названных в его честь. В 1955 году он был приглашен в Йельский университет, где в 1956 году прочитал курс лекций *Silliman Lectures*. Он умер в 1957 году, и его незаконченная работа, написанная на основе этих лекций, была позднее опубликована в виде отдельной книги [1103]. Из этой книги ясно, как много мог бы сделать фон Нейман, если бы остался жив, поскольку он начал осознавать принципиальное отличие мозга от компьютера.

Предметом отдельного исследования в контексте нейронных сетей является создание надежных сетей из нейронов, которые сами по себе считаются ненадежными компонентами. Эта задача была решена в 1956 году фон Нейманом с помощью идеи избыточности, которая была предложена Виноградом и Кованом [163], [1104] в поддержку использования *распределенного избыточного представления* (distributed redundant representation) в нейронных сетях. Эти авторы показали, как большая группа элементов может в совокупности представлять одно понятие при соответствующем повышении робастности и степени параллелизма.

Через 15 лет после выхода классической работы Мак-Каллока и Питца Розенблатт (Rozenblatt) предложил новый подход к задаче распознавания образов, основанный на использовании *персептрона* и нового метода обучения с учителем [902]. Главным достижением этой работы была так называемая *теорема сходимости персептрона* (perceptron convergence theorem), первое доказательство которой было получено Розенблаттом в 1960 году [901]. Другие доказательства этой теоремы были предложены Новиковым [788] и другими учеными. В 1960 году был описан *алгоритм наименьших квадратов* LMS (least mean-square algorithm), который применялся для

построения адаптивных линейных элементов *Adaline* [1131]. Различие между персептроном и моделью Adaline состоит в процедуре обучения. Одной из самых первых обучаемых многослойных нейронных сетей, содержащей многочисленные адаптивные элементы, была структура Madaline (multiply-adaline), предложенная Видроу и его студентами [1137]. В 1967 году для адаптивной классификации образов был использован стохастический градиентный метод [33]. В 1965 году вышла в свет книга Нильсона *Learning Machines* (Обучаемые системы) [786], в которой очень хорошо освещен вопрос *линейной разделимости образов* (linearly separable pattern) с помощью гиперповерхностей. В 1960-е годы (в период господства персептрона) казалось, что нейронные сети позволяют решить практически любую задачу. Однако в 1969 году вышла книга Минского и Пейперта [745], в которой математически строго обоснованы фундаментальные ограничения однослойного персептрона. В небольшом разделе, посвященном многослойным персептронам, утверждалось, что ограничения однослойных персептронов вряд ли удастся преодолеть в их многослойных версиях.

Важной задачей, возникающей при конструировании многослойного персептрона, была названа *проблема присваивания коэффициентов доверия* (credit assignment problem) (т.е. проблема назначения коэффициентов доверия скрытым нейронам сети). Термин *присваивание коэффициентов доверия* впервые использовал Минский в 1961 году [742]. К концу 1960-х уже были сформулированы многие идеи и концепции, необходимые для решения проблемы присваивания коэффициентов доверия для персептрона. Было также разработано множество идей, положенных впоследствии в основу рекуррентных сетей, которые получили название *сетей Хопфилда* (Hopfield network). Однако решения этих важных проблем пришлось ожидать до 1980-х годов. Как отмечено в [222], для такого длительного ожидания существовали объективные причины.

- Одна из причин носила технологический характер: для проведения экспериментов не существовало персональных компьютеров или рабочих станций. Например, когда Габор создавал свой нелинейный обучаемый фильтр, ему и его команде потребовалось шесть лет для того, чтобы создать этот фильтр на аналоговых устройствах [330], [331].
- Другая причина была отчасти психологической, отчасти финансовой. Монография [745] оттолкнула ученых от работ в этом направлении, а научные фонды и агентства перестали обеспечивать его финансовую поддержку.
- Аналогия между нейронными сетями и пространственными решетками (lattice spin) была несовершенной. Более точная модель была создана только в 1975 году [980].

Эти и другие причины способствовали ослаблению интереса к нейронным сетям в 1970-х годах. Многие исследователи (не принимая во внимание психологов и нейробиологов) покинули это поле деятельности на десять лет. Только горстка пионеров

этого направления поддерживала жизнь науки о нейронных сетях. С технологической точки зрения 1970-е годы можно рассматривать как годы застоя.

В 1970-х годах развернулась деятельность в области *карт самоорганизации* (self-organizing map), основанных на *конкурентном принципе обучения* (competitive learning). Принцип самоорганизации впервые был проиллюстрирован с помощью компьютерного моделирования в 1973 году [1100]. В 1976 году была опубликована работа, посвященная картам самоорганизации, отражающим топологически упорядоченную структуру мозга [1159].

В 1980-х годах главный вклад в теорию и конструкцию нейронных сетей был внесен на нескольких фронтах. Этот период был отмечен возобновлением интереса к данному научному направлению.

Гроссберг (Grossberg), ранние работы которого посвящались принципу конкурентного обучения [396–398], в 1980 году открыл новый принцип самоорганизации, получивший название *теории адаптивного резонанса* (adaptive resonance theory) [392]. В основе этой теории лежит использование слоя распознавания "снизу вверх" и слоя генерации "сверху вниз". Если входной и изученный образы совпадают, возникает состояние, называемое *адаптивным резонансом* (т.е. усилением и продлением нейронной активности). Этот *принцип прямой и обратной проекции* (principle of forward/backward projection) был впоследствии снова открыт другими учеными, пришедшими к нему совершенно другим путем.

В 1982 году Хопфилд использовал функцию энергии для описания нового уровня понимания вычислений, выполняемых рекуррентными сетями с симметричными синаптическими связями [480]. Кроме того, он установил изоморфизм между рекуррентной сетью и *изинговской моделью* (Ising model), используемой в статистической физике. Эта аналогия открыла шлюз для притока результатов физической теории (и самих физиков) в нейронное моделирование, трансформировав, таким образом, область нейронных сетей. В 1980-х годах нейронным сетям с обратной связью уделялось большое внимание, и со временем они стали называться *сетями Хопфилда* (Hopfield network). Хотя сети Хопфилда нельзя считать реалистичными моделями нейробиологических систем, в них заложен принцип хранения информации в динамически устойчивых системах. Истоки этого принципа можно найти в более ранних работах других исследователей.

- Крэг и Темперли в 1954–1955 годах сделали следующее наблюдение [228], [229]. Подобно тому, как нейроны могут быть активизированы или приведены в состояние покоя, атомы пространственной решетки могут иметь спины, направленные вверх и вниз .
- В 1967 году Кован ввел "сигмоидальную" характеристику и гладкую функцию активации для нейронов [224].

- Гроссберг в 1967–1968 годах представил *аддитивную модель* (additive model) нейрона, включающую нелинейные разностно-дифференциальные уравнения, и исследовал возможность использования этой модели в качестве основы кратковременной (short-term) памяти [401], [402].
- Амари в 1972 году независимо от других разработал *адаптивную модель* нейрона и использовал ее для изучения динамического поведения нейроноподобных элементов, связанных случайным образом [32].
- Вильсон и Кован в 1972 году вывели системы нелинейных дифференциальных уравнений для описания динамики пространственно-локализованных популяций, содержащих как возбуждающие, так и тормозящие модели нейронов [1161].
- В 1975 году была предложена *вероятностная модель* (probabilistic model) нейрона, которая использовалась для разработки теории кратковременной памяти [662].
- В 1977 году была описана нейросетевая модель, состоящая из простой ассоциативной сети, связанной с нелинейными динамическими элементами [54].

Неудивительно, что работа Хопфилда [480] вызвала лавину дискуссий. Тем не менее принцип хранения информации в динамически устойчивых сетях впервые принял явную форму. Более того, Хопфилд показал, что симметричные синаптические связи гарантируют сходимость к устойчивому состоянию. В 1983 году в [202] выведен общий принцип устойчивости ассоциативной памяти, включающий в качестве частного случая непрерывную версию сети Хопфилда. Отличительной характеристикой аттракторной нейронной сети является естественное включение *времени* в нелинейную динамику сети как важного измерения обучения. В этом контексте теорема Кохена–Гроссберга приобрела особую важность.

Еще одной интересной работой 1982 года стала публикация Кохонена [579], посвященная самоорганизующимся картам, использующим одно- или двухмерную структуру пространственной решетки, которая отличалась от ранней работы [1159]. Модель Кохонена получила более активную поддержку и стала своеобразной точкой отсчета для других инноваций в этой области.

В [560] описана новая процедура, получившая название *моделирования отжига* (simulated annealing), позволяющая решать задачи комбинаторной оптимизации. Имитация отжига уходит корнями в статистическую механику и основана на простейшей идее, впервые использованной в компьютерном моделировании [730]. Идея имитации отжига позднее использовалась при создании стохастической *машины Больцмана* (Boltzmann machine) [9]. Это была первая успешная реализация многослойной нейронной сети. Хотя алгоритм обучения машины Больцмана не обеспечивает такой эффективности, как *алгоритм обратного распространения* (back propagation), он разрушил психологический барьер, показав, что теория Минского и Пейперта [745] была некорректно обоснована. Машина Больцмана также заложила фундамент

для последующей разработки *сигмоидальных сетей доверия* (sigmoid belief network) [778], которые существенно улучшали процесс обучения и обеспечивали связь нейронных сетей с сетями доверия [822]. В [936] описан способ дальнейшего повышения производительности процесса обучения сигмоидальных сетей доверия.

В 1983 году была опубликована работа, посвященная *обучению с подкреплением* (reinforcement learning) [100]. Хотя обучение с подкреплением использовалось и до этого (например, в кандидатской диссертации Минского в 1954 году), эта работа вызвала большой интерес к обучению с подкреплением и его применению в задачах управления. В частности, в этой работе было продемонстрировано, что при использовании обучения с подкреплением можно обеспечить балансировку перевернутого маятника (т.е. шеста, установленного на подвижной платформе) при отсутствии учителя. Системе нужно знать только угол наклона шеста относительно вертикали и момент достижения платформой крайней точки области движения. В 1996 году вышла в свет книга, в которой описывались математические основы обучения с подкреплением, связанные с принципом динамического программирования Беллмана [126].

В 1984 году вышла в свет книга, в которой обосновывается принцип *целенаправленного самоорганизующегося выполнения* (goal-directed self-organized performance), состоящий в том, что понимания сложного процесса легче всего достичь путем синтеза элементарных механизмов, а не анализа "сверху вниз" [149]. Под видом научной фантастики Брайтенберг иллюстрирует этот важный принцип описанием различных систем с простой внутренней архитектурой. Свойства этих систем и их поведение определяются результатами исследования мозга животных. Эту тему автор изучал, явно или опосредованно, более 20 лет.

В 1986 году был разработан *алгоритм обратного распространения ошибки* (back-propagation algorithm) [914]. В том же году издан двухтомник [912]. Эта книга оказала большое влияние на использование алгоритма обучения обратного распространения. Этот алгоритм стал самым популярным для обучения многослойных персептронов. Примерно в это же время, независимо друг от друга, алгоритм обратного распространения был получен и другими исследователями [619], [817]. После открытия алгоритма обратного распространения в середине 1980-х годов оказалось, что он был уже описан ранее в кандидатской диссертации Вербоса (Werbos) в 1974 году в Гарвардском университете. Эта диссертация стала самым первым документированным описанием градиентного метода оптимизации, применяемого к общим моделям сетей, и как частный случай — к моделям нейронных сетей. Основная идея обратного распространения была изложена в [163]. В разделе 2.2 настоящей книги будет описан вариант алгоритма обратного распространения, основанный на формализме Лагранжа. Однако все лавры достались Руммельхарту, Хинтону и Вильямсу [914], [915] за предложение использовать этот алгоритм для машинного обучения и демонстрацию его работы.

В 1988 году Линскер описал новый принцип самоорганизации в перцепционной сети [653]. Этот принцип обеспечивал сохранность максимального количества информации о входных образах за счет ограничений, накладываемых на синаптические связи и динамический диапазон синапса. Аналогичные результаты были получены и несколькими другими исследователями системы зрения независимо друг от друга. Однако лишь концепция Линскера базировалась на теории информации, созданной Шенноном в 1948 году. На ее основе Линскер сформулировал принцип *максимума взаимной информации* (maximum mutual information) Infomax. Его работа открыла двери применению теории информации в нейронных сетях. В частности, применение теории информации к задаче *слепого разделения источников* (blind source separation) [116] послужило примером для применения других информационно-теоретических моделей к решению широкого класса задач так называемого *слепого обращения свертки* (blind deconvolution).

В том же 1988 году Брумхед и Лове описали процедуру построения многослойной сети прямого распространения на базе *радиальных базисных функций* (radial basis function) [160], которая стала альтернативой многослойному персептрону. Положенная в основу такой сети идея радиальных базисных функций уходит корнями к *методу потенциальных функций* (method of potential functions), предложенному в [102]. Теоретические основы метода потенциальных функций были разработаны в [11], [12]. Описание метода функций потенциала приведено также в [269]. Работа [160] вызвала большой интерес и послужила толчком к увеличению объема исследований в области взаимосвязи нейронных сетей и линейных адаптивных фильтров. В 1990 году теория сетей на основе радиальных базисных функций получила дальнейшее развитие за счет применения к ней теории регуляризации Тихонова [847].

В 1989 году вышла книга, в которой описывалось множество различных концепций, заимствованных из нейробиологии и технологии VLSI [720]. Помимо всего прочего, в ней содержались главы, посвященные созданию сетчатки и слуховой улитки на основе кремния. Эти главы явились хорошим примером креативного мышления автора.

В начале 1990-х Вапник (Vapnik) и его коллеги выделили мощный с вычислительной точки зрения класс сетей, обучаемых с учителем, получивший название *машины опорных векторов* (support vector machine). Такие сети позволяют решать задачи распознавания образов, регрессии и оценки плотности [141], [212], [1084], [1085]. Этот новый метод основан на результатах теории обучения на основе выборки конечного размера. Работа систем опорных векторов основана на использовании *VC-измерения (измерения Вапника–Червоненкиса)*, которое позволяет вычислять емкость нейронной сети, обучаемой на множестве примеров [1087], [1088].

В настоящее время хорошо известно, что *хаос* (chaos) является ключевым аспектом многих физических явлений. Возникает вопрос: играет ли хаос столь же важную роль в обучении нейронных сетей? В [310] утверждается, что в биологическом кон-

тексте ответ на этот вопрос является положительным. По мнению автора этой работы, образы нейронной активности не привносятся в мозг извне, а содержатся в нем самом. В частности, хаотическая динамика представляет базис для описания условий, необходимых для проявления свойства эмерджентности в процессе самоорганизации популяций нейронов.

Пожалуй, наибольшее влияние на возобновление интереса к нейронным сетям в 1980-х годах оказали [480] и [912]. За период, прошедший с момента публикации статьи Мак-Каллока и Питца, нейронные сети прошли долгий и тернистый путь. Теория нейронных сетей стала междисциплинарной областью исследований, тесно связанной с нейробиологией, математикой, психологией, физикой и инженерией. Нет необходимости дополнительно говорить о том, что с развитием теории нейронных сетей будут наращивать свой теоретический и прикладной потенциал и эти науки.

Задачи

Модели нейрона

1.1. Диапазон значений логистической функции

$$\varphi(v) = \frac{1}{1 + \exp(-av)}$$

ограничен нулем и единицей. Покажите, что производная функции $\varphi(v)$ описывается выражением

$$\frac{d\varphi}{dv} = a\varphi(v)[1 - \varphi(v)].$$

Каково значение этой производной в начале координат?

1.2. Нечетная сигмоидальная функция задается формулой

$$\varphi(v) = \frac{1 - \exp(-av)}{1 + \exp(-av)} = \tanh(\frac{av}{2}),$$

где tanh обозначает гиперболический тангенс. Областью значений этой функции является интервал от -1 до $+1$. Покажите, что производная функции $\varphi(v)$ описывается выражением

$$\frac{d\varphi}{dv} = \frac{a}{2}[1 - \varphi^2(v)].$$

Каково значение этой производной в начале координат? Предположим, что параметр наклона a — бесконечно большой. В функцию какого вида выродится $\varphi(v)$?

1.3. Рассмотрим еще одну нечетную сигмоидальную функцию (алгебраическую сигмоиду)

$$\varphi(v) = \frac{v}{\sqrt{1+v^2}},$$

значения которой принадлежат интервалу от -1 до $+1$. Покажите, что производная функции $\varphi(v)$ описывается выражением

$$\frac{d\varphi}{dv} = \frac{\varphi^3(v)}{v^3}.$$

Каково значение этой производной в начале координат?

1.4. Рассмотрим следующие функции:

$$\varphi(v) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{v} \exp(-\frac{x^2}{2}) dx \text{ и } \quad \varphi(v) = \frac{2}{\pi} \tan^{-1}(v).$$

Объясните, почему обе эти функции подходят под определение сигмодной функции? Чем они отличаются друг от друга?

1.5. Какие из пяти сигмоидных функций, описанных в задачах 1.1–1.4, можно назвать кумулятивными (вероятностными) функциями распределения? Обоснуйте свой ответ.

1.6. Рассмотрим псевдолинейную функцию активации $\varphi(v)$, показанную на рис. 1.26.

а) Выпишите функциональную зависимость $\varphi(v)$ в аналитическом виде.

б) Как изменится функция $\varphi(v)$, если параметр a принять равным нулю?

1.7. Решите задачу 1.6 для псевдолинейной функции активации, показанной на рис. 1.27.

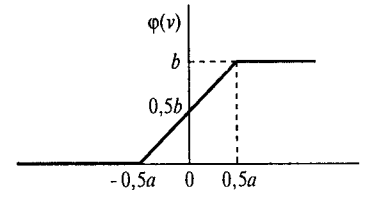


Рис. 1.26. График функции активации

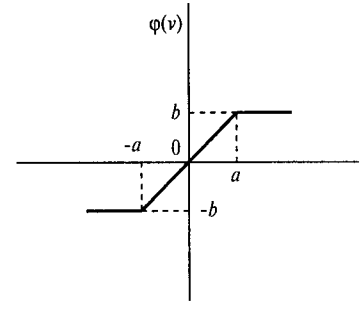


Рис. 1.27. Еще один пример функции активации

1.8. Пусть функция активации нейрона $\varphi(v)$ имеет вид логистической функции из задачи 1.1, где v — индуцированное локальное поле, а параметр наклона a может изменяться. Пусть $x_1, x_2, \ldots, x_m$ — множество входных сигналов, передаваемых на вход нейрона, а b — пороговое значение. Для удобства исключим из рассмотрения параметр a, получив в результате следующую формулу:

$$\varphi(v) = \frac{1}{1 + \exp(-v)}.$$

Как нужно изменить входной сигнал $x_1, x_2, \ldots, x_m$, чтобы получить на выходе прежний сигнал? Обоснуйте свой ответ.

1.9. Нейрон j получает входной сигнал от четырех других нейронов, уровни возбуждения которых равны $10, -20, 4$ и -2. Соответствующие веса связей этого нейрона равны $0,8, 0,2, -1,0$ и $-0,9$. Вычислите выходное значение нейрона j для двух случаев.

а) Нейрон — линейный.

б) Нейрон представлен моделью Мак-Каллока–Питца.

Предполагается, что порог отсутствует.

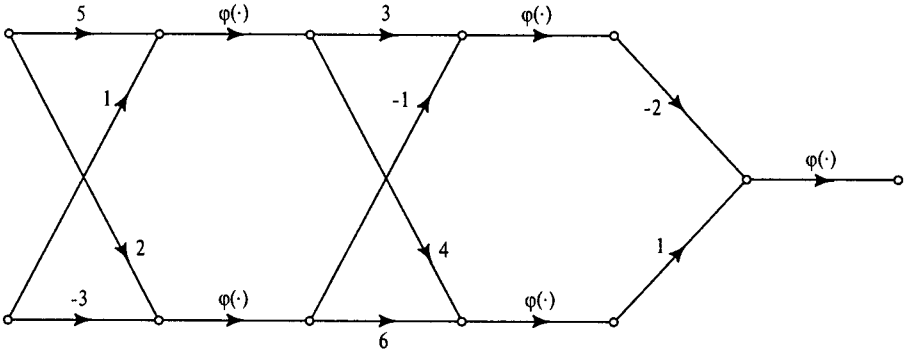


Рис. 1.28. Архитектурный граф сети

1.10. Решите задачу 1.9 для нейрона, модель которого описывается логистической функцией следующего вида:

$$\varphi(v) = \frac{1}{1 + \exp(-v)}.$$

1.11. Решите следующие задачи.

а) Покажите, что формальную модель нейрона Мак-Каллока–Питца можно аппроксимировать сигмоидным нейроном (т.е. нейроном, функция активации которого описывается сигмоидной функцией, а синаптические веса имеют большие значения).

б) Покажите, что линейный нейрон можно аппроксимировать сигмоидным нейроном с маленькими синаптическими весами.

Сетевые архитектуры

1.12. Полносвязная сеть прямого распространения имеет десять входных узлов, два скрытых слоя (один с четырьмя, а другой с пятью нейронами) и один нейрон в выходном слое. Постройте архитектурный граф этой нейронной сети.

1.13. Решите следующие задачи.

а) На рис. 1.28 представлен граф прохождения сигнала по сети прямого распространения вида 2-2-2-1. Функция $\varphi(\cdot)$ является логистической. Опишите отображение вход-выход для этой сети.

б) Пусть выходной нейрон сети, показанной на рис. 1.28, работает в линейной области. Опишите отображение вход-выход для такой сети.

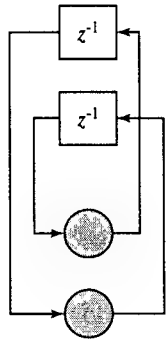


Рис. 1.29. Рекуррентная сеть с двумя нейронами

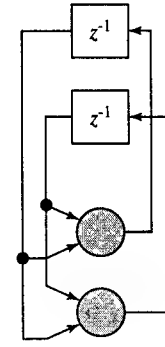


Рис. 1.30. Рекуррентная сеть, содержащая обратные связи нейронов с самими собой

1.14. Сеть, показанная на рис. 1.28, не содержит порогов. Теперь предположим, что эти пороги существуют и равны -1 и $+1$ для верхнего и нижнего нейронов первого скрытого слоя соответственно, а для нейронов второго скрытого слоя — $+1$ и -2 (для верхнего и нижнего нейронов соответственно). Выпишите новую форму отображения вход-выход, определяемого такой сетью.

1.15. Рассмотрим многослойную сеть прямого распространения, все нейроны которой работают в линейной области. Докажите, что такая сеть эквивалентна однослойной сети прямого распространения.

1.16. Сконструируйте полную рекуррентную сеть с пятью нейронами, в которой нейроны не имеют обратных связей сами с собой.

1.17. На рис. 1.29 показан граф передачи сигнала по рекуррентной сети, состоящей из двух нейронов. Выпишите нелинейное разностное уравнение, определяющее эволюцию переменной $x_1(n)$ или $x_2(n)$, которая описывает выход верхнего и нижнего нейронов соответственно. Каков порядок этих уравнений?

1.18. На рис. 1.30 показан граф передачи сигнала по рекуррентной сети с двумя нейронами, каждый из которых имеет обратную связь с самим собой и соседним нейроном. Определите систему из двух нелинейных разностных уравнений первого порядка, описывающих эту сеть.

1.19. Рекуррентная сеть имеет три входных узла, два скрытых и четыре выходных нейрона. Постройте архитектурный граф, описывающий такую сеть.

Представление знаний

1.20. Одна из форм предварительной обработки сигналов основана на *авторегрессионной модели*, описываемой следующим разностным уравнением:

$$y(n) = w_1 y(n-1) + w_2 y(n-2) + \ldots + \quad w_M y(n-M) + v(n), \quad (1.32)$$

где $y(n)$ — выход модели; $v(n)$ — воздействие белого шума с нулевым математическим ожиданием и некоторой предопределенной дисперсией; $w_1, w_2, \ldots, w_M$ — коэффициенты авторегрессионной модели; M — порядок модели. Покажите, что эта модель обладает свойством геометрической инвариантности относительно масштаба и преобразования времени. Как эти две формы инвариантности можно использовать в нейронных сетях?

1.21. Пусть $\mathbf{x}$ — входной вектор, а $\mathbf{s}(\alpha, \mathbf{x})$ — оператор преобразования, зависящий от параметра α. Этот оператор удовлетворяет двум условиям:

- $\mathbf{s}(0, \mathbf{x}) = \mathbf{x}$;
- $\mathbf{s}(\alpha, \mathbf{x})$ — дифференцируемый по α оператор.

Вектор касательной определяется как частная производная $\partial \mathbf{s}(\alpha, \mathbf{x})/\partial\alpha$ [992].

Пусть вектор $\mathbf{x}$ представляет некоторое изображение, а α является параметром поворота. Как вычислить вектор касательной, когда значение α мало? Вектор касательной локально инвариантен относительно угла поворота исходного изображения. Почему?

Процессы обучения

2.1. Введение

Самым важным свойством нейронных сетей является их способность *обучаться* (learn) на основе данных окружающей среды и в результате обучения *повышать* свою производительность. Повышение производительности происходит со временем в соответствии с определенными правилами. Обучение нейронной сети происходит посредством интерактивного процесса корректировки синаптических весов и порогов. В идеальном случае нейронная сеть получает знания об окружающей среде на каждой итерации процесса обучения.

С понятием обучения ассоциируется довольно много видов деятельности, поэтому сложно дать этому процессу однозначное определение. Более того, процесс обучения зависит от точки зрения на него. Именно это делает практически невозможным появление какого-либо точного определения этого понятия. Например, процесс обучения с точки зрения психолога в корне отличается от обучения с точки зрения школьного учителя. Со своей точки зрения (с позиций нейронной сети) мы можем использовать следующее определение, приведенное в [726].

> *Обучение — это процесс, в котором свободные параметры нейронной сети настраиваются посредством моделирования среды, в которую эта сеть встроена. Тип обучения определяется способом подстройки этих параметров.*

Это определение процесса обучения предполагает следующую последовательность событий.

1. В нейронную сеть поступают *стимулы* из внешней среды.
2. В результате этого *изменяются* свободные параметры нейронной сети.
3. После изменения внутренней структуры нейронная сеть *отвечает* на возбуждения уже иным образом.

Вышеуказанный список четких правил решения проблемы обучения называется *алгоритмом обучения*[1] (learning algorithm). Несложно догадаться, что не существует универсального алгоритма обучения, подходящего для всех архитектур нейронных сетей. Существует лишь набор средств, представленный множеством алгоритмов обучения, каждый из которых имеет свои достоинства. Алгоритмы обучения отличаются друг от друга способом настройки синаптических весов нейронов. Еще одной отличительной характеристикой является способ связи обучаемой нейросети с внешним миром. В этом контексте говорят о *парадигме обучения* (learning paradigm), связанной с *моделью* окружающей среды, в которой функционирует данная нейронная сеть.

Структура главы

Эта глава состоит из четырех взаимосвязанных частей. В первой части (разделы 2.2–2.6) речь идет о пяти основных моделей обучения: на основе коррекции ошибок, с использованием памяти, Хеббовском обучении, конкурентном обучении и методе Больцмана. Обучение, основанное на коррекции ошибок, реализует метод оптимальной фильтрации. Обучение на основе памяти предполагает явное использование обучающих данных. Метод Хебба и конкурентный подход к обучению основаны на нейробиологических принципах. В основу метода Больцмана положены идеи статистической механики.

Вторая часть этой главы раскрывает парадигмы обучения. В разделе 2.7 рассматривается задача присваивания коэффициентов доверия (credit assignment), лежащая в основе процесса обучения. Разделы 2.8 и 2.9 содержат обзор двух фундаментальных парадигм обучения: обучения *с* учителем и *без* него.

Третья часть настоящей главы (разделы 2.10–2.12) посвящена решению задач обучения, реализации памяти и адаптации.

Заключительная часть главы (разделы 2.13–2.15) охватывает вероятностные и статистические аспекты процесса обучения. В разделе 2.13 речь идет о дилемме смещения и дисперсии. В разделе 2.14 обсуждается теория статистического обучения, основанная на использовании VC-размерности как меры информационной емкости обучаемой машины. В разделе 2.15 вводится еще одно важное понятие приближенно корректного в вероятностном смысле обучения (probably approximately correct learning), обеспечивающего консервативную модель процесса обучения.

В разделе 2.16 подводятся итоги и приводятся некоторые замечания.

---

[1] Слово "алгоритм" произошло от имени персидского математика Мохаммеда Аль Коварисими (Mohammed Al Kowarisimi), который жил в девятом веке нашей эры. Именно он разработал пошаговые правила сложения, вычитания, умножения и деления действительных десятичных чисел. Когда его имя было записано по-латыни, оно приобрело вид Algorismus, откуда и произошел термин "алгоритм" [420].

2.2. Обучение, основанное на коррекции ошибок

Для того чтобы проиллюстрировать первое правило обучения, рассмотрим простейший случай нейрона k — единственного вычислительного узла выходного слоя нейронной сети прямого распространения (рис. 2.1, *а*). Нейрон k работает под управлением *вектора сигнала* (signal vector) $\mathbf{x}(n)$, производимого одним или несколькими скрытыми слоями нейронов, которые, в свою очередь, получают информацию из входного вектора (возбуждения), передаваемого начальным узлам (входному слою) нейронной сети. Под n подразумевается дискретное время, или, более конкретно, — номер шага итеративного процесса настройки синаптических весов нейрона k. *Выходной сигнал* (output signal) нейрона k обозначается $y_k(n)$. Этот сигнал является единственным выходом нейронной сети. Он будет сравниваться с *желаемым выходом* (desired response), обозначенным $d_k(n)$. В результате получим *сигнал ошибки* (error signal) $e_k(n)$. По определению

$$e_k(n) = d_k(n) - y_k(n). \tag{2.1}$$

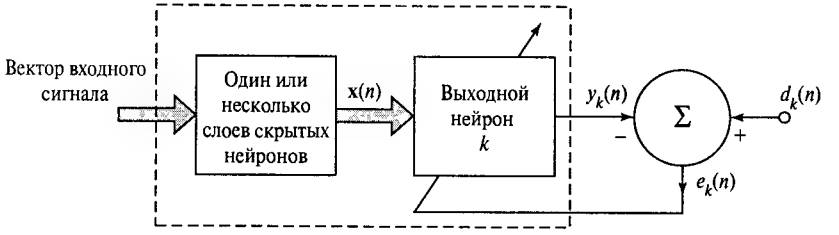
Сигнал ошибки инициализирует *механизм управления* (control mechanism), цель которого заключается в применении последовательности корректировок к синаптическим весам нейрона k. Эти изменения нацелены на пошаговое приближение выходного сигнала $y_k(n)$ к желаемому $d_k(n)$. Эта цель достигается за счет минимизации *функции стоимости* (cost function) или *индекса производительности* (performance index) $\mathbf{E}(n)$, определяемой в терминах сигнала ошибки следующим образом:

$$\mathbf{E}(n) = \frac{1}{2}e_k^2(n), \tag{2.2}$$

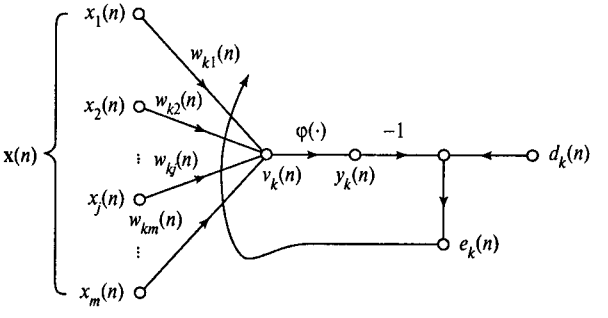
где $\mathbf{E}(n)$ — *текущее значение энергии ошибки* (instantaneous value of the error energy). Пошаговая корректировка синаптических весов нейрона k продолжается до тех пор, пока система не достигнет *устойчивого состояния* (steady state) (т.е. такого, при котором синаптические веса практически стабилизируются). В этой точке процесс обучения останавливается.

Процесс, описанный выше, называется *обучением, основанном на коррекции ошибок* (error-correction learning). Минимизация функции стоимости $\mathbf{E}(n)$ выполняется по так называемому дельта-правилу, или правилу Видроу–Хоффа, названному так в честь его создателей [1141]. Обозначим $w_{kj}(n)$ текущее значение синаптического веса w_{kj} нейрона k, соответствующего элементу $x_j(n)$ вектора $\mathbf{x}(n)$, на шаге дискретизации n. В соответствии с дельта-правилом изменение $\Delta w_{kj}(n)$, применяемое к синаптическому весуw_{kj} на этом шаге дискретизации, задается выражением

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n), \tag{2.3}$$



а) Блочная диаграмма нейронной сети; показаны только нейроны выходного слоя



б) Граф передачи сигнала выходного нейрона

Рис. 2.1. Обучение, основанное на коррекции ошибок

где η — некоторая положительная константа, определяющая *скорость обучения* (rate of learning) и используемая при переходе от одного шага процесса к другому. Из формулы (2.3) видно, что эту константу естественно именовать *параметром скорости обучения* (learning rate parameter). Вербально дельта-правило можно определить следующим образом.

Корректировка, применяемая к синаптическому весу нейрона, пропорциональна произведению сигнала ошибки на входной сигнал, его вызвавший.

Помните, что определенное таким образом дельта-правило предполагает возможность *прямого измерения* (direct measure) сигнала ошибки. Для обеспечения такого измерения требуется поступление желаемого отклика от некоторого внешнего источника, непосредственно доступного для нейрона k. Другими словами, нейрон k должен быть *видимым* (visible) для внешнего мира (рис. 2.1, *а*). На этом рисунке видно, что обучение на основе коррекции ошибки по своей природе является *локальным* (local). Это прямо указывает на то, что корректировка синаптических весов по дельта-правилу может быть локализована в отдельном нейроне k.

Вычислив величину изменения синаптического веса $\Delta w_{kj}(n)$, можно определить его новое значение для следующего шага дискретизации:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n). \tag{2.4}$$

Таким образом, $w_{kj}(n)$ и $w_{kj}(n+1)$ можно рассматривать как *старое* и *новое* значения синаптического веса w_{kj}. В математических терминах можно записать

$$w_{kj}(n) = z^{-1}[w_{kj}(n+1)], \tag{2.5}$$

где z^{-1} — *оператор единичной задержки* (unit delay operator). Другими словами, оператор z^{-1} представляет собой *элемент памяти* (storage element).

На рис. 2.1, *б* представлен граф прохождения сигнала в процессе обучения, основанного на коррекции ошибок, для выделенного нейрона k. Входной сигнал x_k и индуцированное локальное поле v_k нейрона k представлены в виде *предсинаптического* (presynaptic) и *постсинаптического* (postsynaptic) сигналов j-го синапса нейрона k. На рисунке видно, что обучение на основе коррекции ошибок — это пример замкнутой системы с *обратной связью* (closed-loop feedback). Из теории управления известно, что устойчивость такой системы определяется параметрами обратной связи. В данном случае существует всего одна обратная связь, и единственным интересующим нас параметром является коэффициент скорости обучения η. Для обеспечения устойчивости или сходимости итеративного процесса обучения требуется тщательный подбор этого параметра. Выбор параметра скорости обучения влияет также на точность и другие характеристики процесса обучения. Другими словами, параметр скорости обучения η играет ключевую роль в обеспечении производительности процесса обучения на практике.

Обучение, основанное на коррекции ошибок, детально описывается в главе 3 для однослойной сети прямого распространения и в главе 4 — для многослойной.

2.3. Обучение на основе памяти

При *обучении на основе памяти* (memory-based learning) весь прошлый опыт накапливается в большом хранилище правильно классифицированных примеров вида вход-выход: $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, где $\mathbf{x}_i$ — входной вектор, а d_i — соответствующий ему желаемый выходной сигнал. Не ограничивая общности, можно предположить, что выходной сигнал является скаляром. Например, рассмотрим задачу бинарного распознавания образов или классификации на два класса (гипотезы), $\mathbf{C}_1$ и $\mathbf{C}_2$. В этом примере желаемый отклик системы d_i принимает значение 0 (или -1) для класса $\mathbf{C}_1$ и значение $+1$ для класса $\mathbf{C}_2$. Если требуется классифицировать некоторый неизвестный вектор $\mathbf{x}_{\text{test}}$, из базы данных выбирается выход, соответствующий входному сигналу, близкому к $\mathbf{x}_{\text{test}}$.

Все алгоритмы обучения на основе памяти включают в себя две существенные составляющие.

- Критерий, используемый для определения окрестности вектора $\mathbf{x}_{\text{test}}$.
- Правило обучения, применяемое к примеру из окрестности тестового вектора.

Все алгоритмы отличаются друг от друга способом реализации этих двух составляющих.

В простейшем (хотя и эффективном) алгоритме обучения на основе памяти, получившем название *правила ближайшего соседа* (nearest neighbor rule)[2], в окрестность включается пример, ближайший к тестовому. Например, вектор

$$\mathbf{x}'_N \in \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \tag{2.6}$$

считается ближайшим соседом вектора $\mathbf{x}_{\text{test}}$, если выполняется условие

$$\min_i \; d(\mathbf{x}_i, \mathbf{x}_{\text{test}}) = d(\mathbf{x}'_N, \mathbf{x}_{\text{test}}), \tag{2.7}$$

где $d(\mathbf{x}_i, \mathbf{x}_{\text{test}})$ — Евклидово расстояние между векторами $\mathbf{x}_i$ и $\mathbf{x}_{\text{test}}$. Класс, к которому относится ближайший сосед, считается также классом тестируемого вектора $\mathbf{x}_{\text{test}}$. Это правило не зависит от распределения, используемого при генерировании примеров обучения.

В [220] проводится формальное исследование правила ближайшего соседа, применяемого для решения задачи классификации образов. При этом анализ основывается на двух следующих предположениях.

- Классифицируемые примеры $(\mathbf{x}_i, d_i)$ *независимы и равномерно распределены* (independently and identically distributed) в соответствии с совместным распределением примера $(\mathbf{x}, d)$.
- Размерность обучающего множества N бесконечно велика.

Показано, что при этих двух предположениях вероятность ошибки классификации при использовании правила ближайшего соседа вдвое превышает *байесовскую вероятность ошибки* (Bayes probability error). (Байесовская вероятность ошибки — это минимальная вероятность ошибки на множестве всех правил принятия решения.) Байесовская вероятность ошибки описывается в главе 3. В этом контексте можно считать, что половина классификационной информации для обучающего множества бесконечного размера содержится в данных о ближайшем соседе. Это довольно неожиданный результат.

---

[2] Принципу ближайшего соседа посвящено очень много книг, например [297].

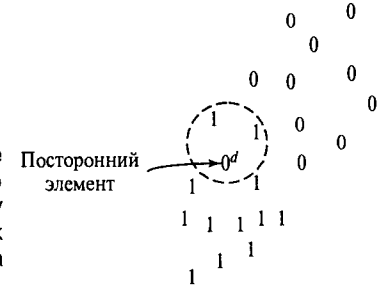


Рис. 2.2. Область в штриховой окружности содержит две точки, принадлежащие классу 1, и одну, принадлежащую классу 0. Точка d соответствует тестируемому вектору $\mathbf{x}_{\text{test}}$. При $k = 3$ классификатор на основе k-ближайших соседей отнесет точку d к классу 1, несмотря на то, что она лежит ближе всего к "выбросу", относящемуся к классу 0

Вариацией классификатора на основе ближайшего соседа является *классификатор k-ближайших соседей* (k-nearest neighbor classifier). Он описывается следующим образом.

Находим k классифицированных соседей, ближайших к вектору $\mathbf{x}_{test}$, где k — некоторое целое число.

Вектор $\mathbf{x}_{\text{test}}$ относим к тому классу (гипотезе), который чаще других встречается среди k-ближайших соседей тестируемого вектора.

Таким образом, классификатор на основе k-ближайших соседей работает подобно устройству усреднения. Например, он может не учесть единичный "выброс", как показано на рис. 2.2 для $k = 3$. *Выброс* (outlier) — это наблюдение, которое отличается от номинальной модели.

В главе 5 рассматривается еще один тип классификатора на основе памяти, который называется *сетью на базе радиальных базисных функций.*

2.4. Обучение Хебба

Постулат обучения Хебба (Hebb's postulate of learning) является самым старым и самым известным среди всех правил обучения. Он назван в честь нейрофизиолога Хебба. Приведем цитату из его книги [445].

> *Если аксон клетки А находится на достаточно близком расстоянии от клетки В и постоянно или периодически участвует в ее возбуждении, наблюдается процесс метаболических изменений в одном или обоих нейронах, выражающийся в том, что эффективность нейрона А как одного из возбудителей нейрона В возрастает.*

Хебб предложил положить это наблюдение в основу процесса ассоциативного обучения (на клеточном уровне). По его мнению, это должно было привести к постоянной модификации шаблона активности пространственно-распределенного "ансамбля нервных клеток".

Это утверждение было сделано в нейробиологическом контексте, но его можно перефразировать в следующее правило, состоящее из двух частей [180], [1016].

1. *Если два нейрона по обе стороны синапса (соединения) активизируются одновременно (т.е. синхронно), то прочность этого соединения возрастает.*
2. *Если два нейрона по обе стороны синапса активизируются асинхронно, то такой синапс ослабляется или вообще отмирает.*

Функционирующий таким образом синапс называется *синапсом Хебба*[3] (Hebbian synapse) (обратите внимание, что исходное правило Хебба не содержало второй части последнего утверждения). Если быть более точным, то синапс Хебба использует *зависящий от времени, в высшей степени локальный механизм взаимодействия, изменяющий эффективность синаптического соединения в зависимости от корреляции между предсинаптической и постсинаптической активностью.* Из этого определения можно вывести следующие четыре свойства (ключевых механизма), характеризующие синапс Хебба [161].

1. *Зависимость от времени* (time-dependent mechanism). Синапс Хебба зависит от точного времени возникновения предсинаптического и постсинаптического сигналов.
2. *Локальность* (local mechanism). По своей природе синапс является узлом передачи данных, в котором информационные сигналы (представляющие текущую активность предсинаптических и постсинаптических элементов) находятся в *пространственно-временной* (spatiotemporal) близости. Эта локальная информация используется синапсом Хебба для выполнения локальных синаптических модификаций, характерных для данного входного сигнала.
3. *Интерактивность* (interactive mechanism). Изменения в синапсе Хебба определяются сигналами на обоих его концах. Это значит, что форма обучения Хебба зависит от степени взаимодействия предсинаптического и постсинаптического сигналов (предсказание нельзя построить на основе только одного из этих сигналов). Заметим, что такая зависимость или интерактивность может носить детерминированный или статистический характер.
4. *Корреляция* (correlational mechanism). Одна из интерпретаций постулата обучения Хебба состоит в том, что условием изменения эффективности синаптической связи является зависимость между предсинаптическим и постсинаптическим сигналами. В соответствии с этой интерпретацией для модификации синапса необходимо обеспечить одновременность предсинаптического и постсинаптического

[3] Подробный обзор синапсов Хебба, включающий и историческую справку, содержится в [161] и [316], а также в [207].

сигналов. Именно по этой причине синапс Хебба иногда называют *конъюнктивным синапсом* (conjunctional synapse). Другая интерпретация постулата обучения Хебба использует характерные для синапса Хебба механизмы взаимодействия в статистических терминах. В частности, синаптические изменения определяются корреляцией предсинаптического и постсинаптического сигналов во времени. Учитывая это, синапс Хебба иногда называют *корреляционным синапсом* (correlational synapse). Сама корреляция является основой обучения [279].

Усиление и ослабление синаптической связи

Приведенное выше определение синапса Хебба не учитывает дополнительные процессы, которые могут привести к ослаблению синаптической связи между парой нейронов. Исходя из этого, можно несколько обобщить концепцию модификации связей Хебба, используя тот факт, что положительно коррелированное функционирование приводит к усилению синаптической связи, а отрицательная корреляция или ее отсутствие — к ее ослаблению [1016]. Ослабление синаптической связи может также носить и не интерактивный характер. В частности, условие взаимодействия для ослабления синаптической связи может носить случайный характер и не зависеть от предсинаптического и постсинаптического сигналов.

Систематизируя эти определения, можно выделить следующие модели модификации синаптических связей: *хеббовскую* (Hebbian), *антихеббовскую* (anti-Hebbian) и *нехеббовскую* (non-Hebbian) [810]. Следуя этой схеме, можно сказать, что синаптическая связь по модели Хебба усиливается при положительной корреляции предсинаптических и постсинаптических сигналов и ослабляется в противном случае. И, наоборот, согласно антихеббовской модели, синаптическая связь ослабляется при положительной корреляции предсинаптического и постсинаптического сигналов и усиливается в противном случае. Однако в обеих моделях изменение синаптической эффективности основано на зависящем от времени в высшей степени локальном и интерактивном по своей природе механизме. В этом контексте антихеббовская модель все равно остается хеббовской по природе, но не по функциональности. Не-хеббовская модель вообще не связана с механизмом модификации, предложенным Хеббом.

Математические модели предложенного Хеббом механизма модификации синаптической связи

Для того чтобы описать обучение Хебба в математических терминах, рассмотрим синаптический вес w_{kj} нейрона k с предсинаптическим и постсинаптическим сигналами x_j и y_k. Изменение синаптического веса w_{kj} в момент времени n можно выразить следующим соотношением:

$$\Delta w_{kj}(n) = F(y_k(n), x_j(n)), \tag{2.8}$$

где $F(\cdot,\cdot)$ — некоторая функция, зависящая от предсинаптического и постсинаптического сигналов. Сигналы $x_j(n)$ и $y_k(n)$ часто рассматривают без учета размерности. Формула (2.8) может быть записана в разных формах, каждая из которых все равно считается хеббовской. Рассмотрим только две следующие формы.

Гипотеза Хебба

Простейшая форма обучения Хебба имеет следующий вид:

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n), \tag{2.9}$$

где η — положительная константа, определяющая *скорость обучения* (rate of learning). Выражение (2.9) ясно подчеркивает корреляционную природу синапса Хебба. Ее иногда называют *правилом умножения активности* (activity product rule). Верхний график на рис. 2.3 графически иллюстрирует формулу (2.9). Он представляет зависимость изменения $\Delta w_{kj}(n)$ от выходного сигнала (постсинаптической активности) $y_k(n)$. На рисунке видно, что регулярное приложение входного сигнала (предсинаптической активности) $x_j(n)$ ведет к увеличению $y_k(n)$ и *экспоненциальному росту* активности (exponential growth), приводящему к насыщению синаптической связи. В этой точке синапс уже не содержит никакой информации, и избирательность связей теряется.

Гипотеза ковариации

Одним из способов преодоления ограничений гипотезы Хебба является использование *гипотезы ковариации* (covariance hypothesis), предложенной в [956], [957]. Согласно этой гипотезе, предсинаптический и постсинаптический сигналы в формуле (2.9) заменяются отклонениями этих сигналов от их средних значений на данном отрезке времени. Обозначим символами $\bar{x}$ и $\bar{y}$ *усредненные по времени значения* предсинаптического x_j и постсинаптического y_k сигналов. Согласно гипотезе ковариации, изменение синаптического веса w_{kj} вычисляется по формуле

$$\Delta w_{kj} = \eta(x_j - \bar{x})(y_k - \bar{y}), \tag{2.10}$$

где η — параметр скорости обучения. Средние значения x и y содержат предсинаптический и постсинаптический пороги, определяющие знак синаптической модификации. В частности, гипотеза ковариации обеспечивает следующее.

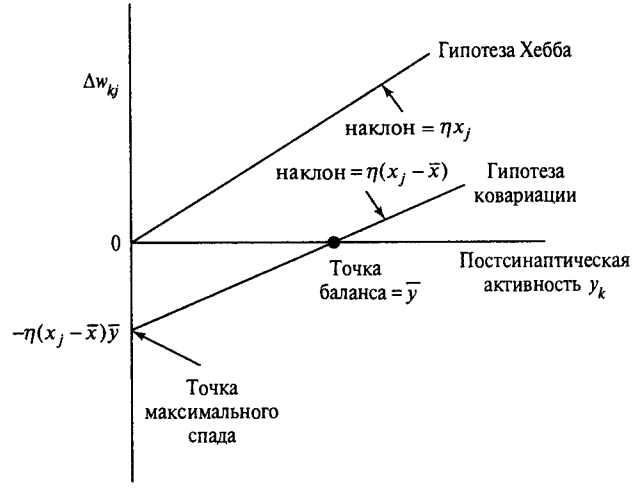


Рис. 2.3. Иллюстрация гипотез Хебба и ковариации

- Сходимость к нетривиальному состоянию, которое достигается при $x_k = \bar{x}$ и $y_j = \bar{y}$.
- Прогнозирование *усиления* (potentiation) и *ослабления* (depression) синаптической связи.

На рис. 2.3 показано отличие гипотезы Хебба от гипотезы ковариации. В обоих случаях зависимость величины Δw_{kj} от y_k является линейной, однако пересечение с осью y_k для гипотезы Хебба происходит в начале координат, а для гипотезы ковариации — в точке $y_k = \bar{y}$.

При внимательном исследовании выражения (2.10) можно сделать следующие выводы.

1. Синаптический вес связи усиливается при высоком уровне предсинаптического и постсинаптического сигналов, т.е. в том случае, когда одновременно удовлетворяются следующие условия: $x_j > \bar{x}$ и $y_k > \bar{y}$.
2. Синаптический вес ослабляется в следующих ситуациях.
   - Предсинаптическая активность (т.е. $x_j > \bar{x}$) не вызывает существенной постсинаптической активности ($y_k < \bar{y}$).
   - Постсинаптическая активность ($y_k > \bar{y}$) возникает при отсутствии существенной предсинаптической активности ($x_j < \bar{x}$).

Такое поведение можно рассматривать как форму временной конкуренции между входными моделями.

Существует строгое физиологическое доказательство[4] реализации принципа обучения Хебба в области мозга, называемой *гипокампусом* (hippocampus). Эта область играет важную роль в некоторых аспектах обучения и запоминания. Физиологические результаты еще раз подтверждают правильность постулата обучения Хебба.

---

[4] **Долговременное усиление — физиологическое доказательство модели синапса Хебба.**

Хебб еще в 1949 году изложил свое видение механизмов синаптической памяти [445], но прошло еще около четверти столетия, пока были получены его экспериментальные доказательства. В 1973 году была опубликована работа, описывающая модификации синаптических связей, вызванные внешним возбуждением в области головного мозга, называемой *гипокампусом* (hippocampus) [133]. Авторы работы приложили импульсы электрического возбуждения к главному тракту, направленному в эту структуру, и записали синаптические отклики. Пока эта структура функционировала автономно, она характеризовалась устойчивой морфологией базовых откликов — короткими высокочастотными сериями импульсов в том же тракте. После применения возбуждающих тестовых импульсов амплитуда отклика увеличилась. Внимание исследователей привлек тот факт, что это увеличение амплитуды довольно продолжительное время не затухало. Это явление было названо *долговременным усилением* (long-term potentiation — LTP).

В настоящее время ежегодно публикуется множество работ, посвященных явлению LTP, и многое уже известно о механизмах его реализации. Например, известно, что эффект усиления ограничен только *возбужденными трактами* (pathway). Известно также, что механизм LTP обладает ассоциативными свойствами, под которыми подразумевается проявление эффекта взаимодействия трактов, активизируемых совместно. Например, если слабое возбуждение, не вызывающее эффекта LTP, применять в паре с сильным, то слабый сигнал также будет усиливаться. Этот эффект называют *ассоциативным свойством*, так он аналогичен ассоциативным свойствам систем обучения. В экспериментах Павлова, посвященных изучению условного рефлекса, нейтральный звуковой возбудитель (слабый) применялся в паре с сильным (еда). Результатом стало появление условного рефлекса (conditioned response) — выделение слюны в ответ на данный звуковой сигнал.

Большинство экспериментальных работ в этой области было сфокусировано на ассоциативных свойствах LTP. В большинстве синапсов, поддерживающих LTP, в качестве нейропередатчика используется глутамат (glutamate). В постсинаптических нейронах существует множество различных рецепторов, реагирующих на глутамат. Все эти рецепторы имеют разные свойства, но мы остановимся только на двух из них. Главный синаптический отклик инициируется возбуждением рецептора AMPA (этот рецептор был назван в честь наркотика, на который реагирует наиболее явно). Когда в эксперименте с LTP записывался результат, он в первую очередь относился на счет возбуждения рецептора AMPA. После синаптической активации высвобождался глутамат, который связывался с рецепторами в постсинаптической мембране. Каналы ионов, являющиеся частью рецепторов AMPA, открывались, передавая поток, являющийся основой синаптического отклика.

Второй тип глутаматых рецепторов — NMDA — тоже имеет ряд интересных свойств. Для открытия ассоциированного ионного канала в рецепторе NMDA не достаточно глутаматного связывания. Этот канал остается заблокированным, пока в процессе синаптической активности (в том числе и рецепторами AMPA) не будет сгенерирован большой скачок напряжения. Следовательно, если рецепторы AMPA являются химически зависимыми, то рецепторы NMDA являются как химически, так и электрически зависимыми. Канал, ассоциированный с рецептором AMPA, связан с движением ионов соды (которые вызывают синаптический ток). Канал, связанный с рецептором NMDA, обеспечивает попадание кальция в клетку. Хотя движение кальция влияет также и на *мембранный ток* (membrane current), его главной ролью является порождение цепочки событий, увеличивающих продолжительность реакции, связанной с рецептором AMPA.

Так действует механизм синапса, описанный Хеббом. Рецептор NMDA требует как предсинаптической (высвобождение глутамата), так и постсинаптической активности. Как этого добиться? Наличием достаточно сильного входного сигнала. Когда слабый входной сигнал одного синапса интегрируется с сильным сигналом другого, первый синапс высвобождает свой глутамат, а второй обеспечивает достаточно большой скачок напряжения, достаточный для активации рецепторов NMDA, связанных со слабым синапсом.

Несмотря на то что исходное утверждение Хебба относится к однонаправленному обучению, правило двунаправленного обучения обеспечивает более высокую степень гибкости нейронных сетей. При его использовании синаптические веса отдельных нейронов могут не только увеличиваться, но и уменьшаться. Следует знать, что существуют экспериментальные доказательства существования механизмов синаптического ослабления. Если слабый входной сигнал активизируется без сопутствующего ему сильного, синаптический вес, как правило, ослабляется. Этот результат можно наблюдать в виде реакции синаптической системы на низкочастотное воздействие. Такое явление называется *долговременным торможением* (long-term depression, или LTD). Существует также подтверждение *гетеросинаптического торможения* (geterosynaptic depression). Если LTD — это торможение, влияющее на активизированный вход, то влияние гетеросинаптического торможения распространяется только на не активизированные входы.

2.5. Конкурентное обучение

Как следует из самого названия, в *конкурентном обучении* (competitive learning)[5] выходные нейроны нейронной сети конкурируют между собой за право быть активизированными. Если в нейронной сети, основанной на обучении Хебба, одновременно в возбужденном состоянии может находиться несколько нейронов, то в конкурентной сети в каждый момент времени может быть активным только один нейрон. Благодаря этому свойству конкурентное обучение очень удобно использовать для изучения статистических свойств, используемых в задачах классификации входных образов.

Правило конкурентного обучения основано на использовании трех основных элементов [913].

- Множество одинаковых нейронов со случайно распределенными синаптическими весами, приводящими к *различной* реакции нейронов на один и тот же входной сигнал.
- *Предельное значение* (limit) "силы" каждого нейрона.
- Механизм, позволяющий нейронам *конкурировать* за право отклика на данное подмножество входных сигналов и определяющий единственный активный выходной нейрон (или по одному нейрону на группу). Нейрон, победивший в этом соревновании, называют *нейроном-победителем*, а принцип конкурентного обучения формулируют в виде лозунга "победитель забирает все" (*winner-takes-all*).

Таким образом, каждый отдельный нейрон сети соответствует группе близких образов. При этом нейроны становятся *детекторами признаков* (feature detector) различных классов входных образов.

Простейшая нейронная сеть с конкурентным обучением содержит единственный слой выходных нейронов, каждый из которых соединен с входными узлами. В такой сети могут существовать обратные связи между нейронами (рис. 2.4). В подобной архитектуре обратная связь обеспечивает *латеральное торможение* (lateral inhibition)[6], когда каждый нейрон стремится "затормозить" связанные с ним нейроны.

---

[5] Идею конкурентного обучения можно проследить еще в ранних работах, в том числе в [1100], посвященной самоорганизации чувствительных к ориентации нервных клеток в извилинах коры головного мозга (striate cortex); в [327], посвященной самоорганизующимся многоуровневым нейронным сетям, получившим название *неокогнитронов*; в [1159], посвященной формированию образных нейронных связей (patterned neural connection) с помощью самоорганизации; и в [396–398], посвященных адаптивной классификации образов. Впоследствии было доказано, что конкурентное обучение играет важную роль при формировании топографических отображений в структуре мозга [272], а недавняя экспериментальная работа обеспечивает дальнейшую физиологическую верификацию конкурентного обучения [41].

[6] Латеральное торможение (см. рис. 2.4) присуще также нейробиологическим системам. Множество сенсориых тканей (в частности, сетчатка глаза, ушная раковина и осязательные нервные окончания на коже) организовано таким образом, что воздействие на любой участок приводит к торможению соседних нервных клеток [66], [295]. В системе осязания человека латеральное торможение проявляется в виде явления, связанного с *полосами Маха* (Mach band), получившими свое название в честь известного физика Эрнста Маха [694].

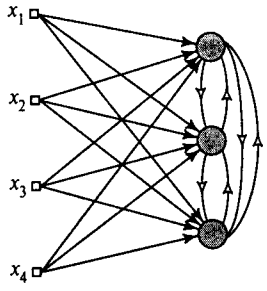


Рис. 2.4. Архитектурный граф простой сети конкурентного обучения с прямыми (возбуждающими) связями от входных узлов к нейронам и обратными (тормозящими) связями между нейронами (последние обозначены на рисунке незаштрихованными стрелками)

Прямые синаптические связи в сети, показанной на рис. 2.4, являются *возбуждающими* (excitatory).

Для того чтобы нейрон k победил в конкурентной борьбе, его индуцированное локальное поле v_k для заданного входного образа $\mathbf{x}$ должно быть максимальным среди всех нейронов сети. Тогда выходной сигнал y_k нейрона-победителя k принимается равным единице. Выходные сигналы остальных нейронов при этом устанавливаются в значение нуль. Таким образом, можно записать:

$$y_k = \begin{cases} 1, & \text{если } v_k > v_j \text{ для всех } \ j, j \neq k, \\ 0 & \text{в остальных случаях,} \end{cases} \tag{2.11}$$

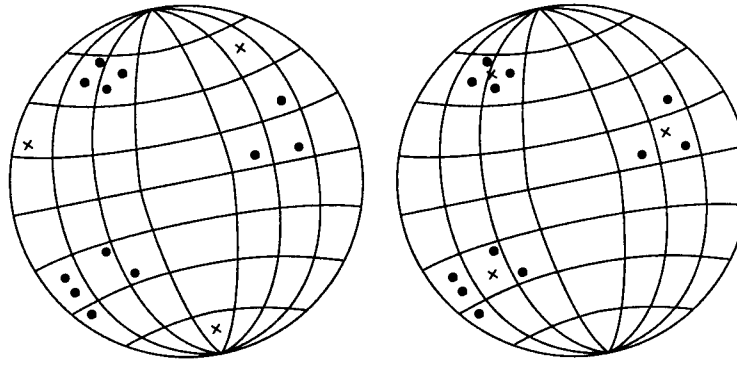
где индуцированное локальное поле v_k представляет сводное возбуждение нейрона k от всех входных сигналов и сигналов обратной связи.

Пусть w_{kj} — синаптический вес связи входного узла j с нейроном k. Предположим, что синаптические веса всех нейронов фиксированы (т.е. положительны), при этом

$$\sum_j w_{kj} = 1 \ \text{ для всех } \ k. \tag{2.12}$$

Тогда обучение этого нейрона состоит в смещении синаптических весов от неактивных к активным входным узлам. Если нейрон не формирует отклика на конкретный входной образ, то он и не обучается. Если некоторый нейрон выигрывает в конкурентной борьбе, то веса связей этого нейрона равномерно распределяются между его активными входными узлами, а связи с неактивными входными узлами

Например, если посмотреть на лист белой бумаги, наполовину закрашенный черным цветом, можно увидеть линии, параллельные границе разделения, которые имеют цвет "более белого" с белой стороны и "чернее черного" с черной, даже если заливка половинок абсолютно монотонна. Полосы Маха физически не существуют — они являются всего лишь зрительной иллюзией, представляющей собой "завышение" и "занижение" сигнала, вызванные функцией дифференциации (differentiating) латерального торможения.



а) б)

Рис. 2.5. Геометрическая интерпретация процесса конкурентного обучения. Точками представлены входные векторы, а крестиками — векторы синаптических весов трех выходных нейронов в исходном (а) и конечном (б) состоянии сети

ослабляются. Согласно *правилу конкурентного обучения* (competitive learning rule) изменение Δw_{kj} синаптического весаw_{kj} определяется следующим выражением:

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}), & \text{если нейрон } k \text{ побеждает в соревновании,} \\ 0, & \text{если нейрон } k \text{ побеждает в соревновании,} \end{cases} \tag{2.13}$$

где η — параметр скорости обучения. Это правило отражает смещение вектора синаптического веса $\mathbf{w}_k$ победившего нейрона k в сторону входного образа $\mathbf{x}$.

Для иллюстрации сущности конкурентного обучения можно использовать геометрическую аналогию (рис. 2.5) [913]. Предполагается, что все входные образы (векторы) $\mathbf{x}$ имеют некоторую постоянную Евклидову норму. Таким образом, их можно изобразить в виде точек на N-мерной единичной сфере, где N — количество входных узлов. N также является размерностью вектора синаптических весов $\mathbf{w}_k$. Предполагается, что все нейроны сети имеют ту же Евклидову длину (норму), т.е.

$$\sum_j w_{kj}^2 = 1 \quad \text{для всех } k. \tag{2.14}$$

Если синаптические веса правильно масштабированы, они формируют набор векторов, которые проецируются на ту же N-мерную единичную сферу. На рис. 2.5, *а* можно выделить три естественные группы (кластеры) точек, представляющие входные образы. На этом рисунке также показано вероятное начальное состояние сети (отмечено крестиками) до начала обучения. На рис. 2.5, *б* показано конечное состояние сети, полученное в результате конкурентного обучения, в котором синаптические веса каждого выходного нейрона смещены к центрам тяжести соответствующих кла-

стеров [453], [913]. На этом примере продемонстрирована способность нейронной сети решать задачи *кластеризации* в процессе конкурентного обучения. Однако для "устойчивого" решения этой задачи входные образы должны формировать достаточно разрозненные группы векторов. В противном случае сеть может стать неустойчивой, поскольку в ответ на заданный входной образ будут формироваться отклики от различных выходных нейронов.

2.6. Обучение Больцмана

Правило обучения Больцмана, названное так в честь Людвига Больцмана, представляет собой стохастический алгоритм обучения, основанный на идеях стохастической механики[7]. Нейронная сеть, созданная на основе обучения Больцмана, получила название *машины Больцмана* (Boltzmann machine) [9], [464].

В машине Больцмана все нейроны представляются рекуррентными структурами, работающими с бинарными сигналами. Это значит, что они могут находиться во включенном (соответствующем значению +1) или выключенном (соответствующем значению -1) состоянии. Такая машина характеризуется функцией энергии E, значение которой определяется конкретными состояниями отдельных нейронов, составляющих эту машину. Это можно описать следующим выражением:

$$E = -\frac{1}{2}\sum_{j}\sum_{k(j\neq k)} w_{kj}x_k x_j, \tag{2.15}$$

где x_j — состояние нейрона j; w_{kj} — синаптический вес связи нейронов j и k. Условие $j \neq k$ подчеркивает тот факт, что в этой сети нейроны не имеют обратных связей с самими собой. Работа этой машины заключается в случайном выборе некоторого нейрона (предположим, k-го) на определенном шаге процесса обучения и переводе этого нейрона из состояния x_k в состояние $-x_k$ при некоторой температуре T с вероятностью

---

[7] Важность статистической термодинамики при изучении вычислительных машин была доказана еще Джоном фон Нейманом в третьей из пяти лекций по Теории и организации вычислительных автоматов, проведенных в университете штата Иллинойс в 1949 году. В своей третьей лекции на тему "Статистические теории информации" фон Нейман сказал следующее.

"Понятия термодинамики обязательно войдут в новую теорию информации. Существуют признаки того, что информация подобна энтропии и процессы энтропии подобны процессам в обработке информации. Чаще всего не удается определить функцию автомата или его эффективность, не охарактеризовав среду, в которой он работает, с помощью статистических свойств, подобных тем, которые характеризуют среду в термодинамике. Статистические свойства среды автомата, естественно, сложнее, чем стандартные термодинамические переменные температуры, однако по сути они очень близки".

$$P(x_k \to -x_k) = \frac{1}{1 + \exp(-\Delta E_k / T)}, \tag{2.16}$$

где ΔE_k — *изменение энергии машины* (т.е. изменение ее функции энергии), вызванное переходом состояний. Заметим, что под обозначением T скрывается не физическая температура, а *псевдотемпература* (pseudotemperature), определение которой было приведено в главе 1. При многократном применении этого правила машина достигает *термального равновесия* (thermal equilibrium).

Нейроны машины Больцмана можно разбить на две функциональные группы: (visible) и *скрытые* (hidden). Видимые нейроны реализуют интерфейс между сетью и средой ее функционирования, а скрытые работают независимо от внешней среды. Рассмотрим два режима функционирования такой сети.

- *Скованное состояние* (clamped condition), в котором все видимые нейроны находятся в состояниях, предопределенных внешней средой.
- *Свободное состояние* (free-running condition), в котором все нейроны (как видимые, так и скрытые) могут свободно функционировать.

Обозначим ρ_{kj}^{+} *корреляцию* между состояниями нейронов j и k в скованном состоянии. Аналогично, ρ_{kj}^{-} обозначим *корреляцию* (correlation) между состояниями нейронов j и k, когда сеть находится в свободном состоянии. Обе эти корреляции усредняются по всем возможным состояниям машины, находящейся в условиях термального равновесии. Затем, согласно *правилу обучения Больцмана* (Boltzmann learning rule), изменение Δw_{kj} синаптического весаw_{kj} связи между нейронами k и j определяется следующим выражением [464]:

$$\Delta w_{kj} = \eta(\rho_{kj}^{+} - \rho_{kj}^{-}), \qquad j \neq k, \tag{2.17}$$

где η — параметр скорости обучения. Заметим, что значения ρ_{kj}^{+} и ρ_{kj}^{-} изменяются в диапазоне от -1 до $+1$.

Краткий обзор методов статистической механики будет представлен вниманию читателя в главе 11. Там же мы глубже рассмотрим машину Больцмана и некоторые другие стохастические машины.

2.7. Задача присваивания коэффициентов доверия

При изучении алгоритмов обучения распределенных систем полезно ознакомиться с концепцией *присваивания коэффициентов доверия* (credit assignment) [742]. По существу это задача присваивания коэффициентов доверия или недоверия всем результатам, полученным некоторой обучаемой машиной. (Задачу *присваивания коэффициентов доверия* иногда называют *задачей загрузки* (loading problem), т.е. распределения множества обучающих данных по свободным параметрам сети.)

Во многих случаях зависимость выходов от внутренних решений определяется последовательностью действий, выполняемых обучаемой машиной. Другими словами, внутренние решения влияют на выполнение конкретных действий, после чего именно эти действия, а не сами решения непосредственно определяют общие результаты. В такой ситуации можно выполнить декомпозицию задачи присваивания коэффициентов доверия на две другие подзадачи [1035].

1. Присваивание коэффициентов доверия результатам действий. Эта задача называется *временной задачей присваивания коэффициентов доверия* (temporal credit-assignment problem). В ней определяется промежуток времени, в течение которого реально выполняются действия, которым отпущен кредит доверия.
2. Присваивание коэффициентов доверия действий внутренним решениям. Это называется *структурной задачей присваивания коэффициентов доверия* (structural credit-assignment problem). В ней коэффициенты доверия назначаются *внутренним структурам* (internal structures) действий, генерируемых системой.

Структурная задача присваивания коэффициентов доверия имеет смысл в контексте многокомпонентных обучаемых машин, когда необходимо точно определить, поведение какого элемента системы нужно скорректировать и на какую величину, чтобы повысить общую производительность системы. С другой стороны, временная задача присваивания коэффициентов доверия ставится в том случае, когда обучаемая машина выполняет достаточно много действий, приводящих к некоторому результату, и требуется определить, какие из этих действий несут ответственность за результат. Сочетание временной и структурной задач присваивания коэффициентов доверия требует усложнения поведения распределенной обучаемой машины [1153].

Например, задача присваивания коэффициентов доверия возникает в том случае, когда обучение на основе коррекции ошибок применяется к многослойным нейронным сетям прямого распространения. Действия каждого скрытого и выходного нейрона такой сети важны для формирования правильного результата в данной прикладной области. Это значит, что для решения поставленной задачи необходимо задать определенные формы поведения всех нейронов в процессе обучения на основе коррекции ошибок. В этом контексте вернемся к рис. 2.1, *а*. Поскольку выходной нейрон

k является видимым внешнему миру, желаемый выходной сигнал можно направить непосредственно на этот нейрон. Так как мы рассматриваем выходной нейрон, метод обучения на основе коррекции ошибок можно применить непосредственно к нему (см. раздел 2.2). Но как назначить коэффициенты доверия или недоверия действиям скрытых нейронов, если процесс обучения на основе коррекции ошибок применяется к их же синаптическим весам? Ответ на этот фундаментальный вопрос требует более детального описания, которое и будет приведено в главе 4, в которой детально рассматриваются алгоритмы построения многослойных нейронных сетей прямого распространения.

2.8. Обучение с учителем

Рассмотрим парадигмы обучения нейронных сетей. Начнем с парадигмы *обучения с учителем* (supervised learning). На рис. 2.6 показана блочная диаграмма, иллюстрирующая эту форму обучения. Концептуально участие учителя можно рассматривать как наличие знаний об окружающей среде, представленных в виде пар *вход-выход*. При этом сама среда *неизвестна* обучаемой нейронной сети. Теперь предположим, что учителю и обучаемой сети подается обучающий вектор из окружающей среды. На основе встроенных знаний учитель может сформировать и передать обучаемой нейронной сети желаемый отклик, соответствующий данному входному вектору. Этот желаемый результат представляет собой оптимальные действия, которые должна выполнить нейронная сеть. Параметры сети корректируются с учетом обучающего вектора и сигнала ошибки. *Сигнал ошибки* (error signal) — это разность между желаемым сигналом и текущим откликом нейронной сети. Корректировка параметров выполняется пошагово с целью *имитации* (emulation) нейронной сетью поведения учителя. Эта эмуляция в некотором статистическом смысле должна быть оптимальной. Таким образом, в процессе обучения знания учителя передаются в сеть в максимально полном объеме. После окончания обучения учителя можно отключить и позволить нейронной сети работать со средой самостоятельно.

Описанная форма обучения с учителем является ничем иным, как обучением на основе коррекции ошибок, описанным в разделе 2.2. Это замкнутая система с обратной связью, которая не включает в себя окружающую среду. Производительность такой системы можно оценивать в терминах среднеквадратической ошибки или суммы квадратов ошибок на обучающей выборке, представленной в виде функции от свободных параметров системы. Для такой функции можно построить многомерную *поверхность ошибки* (error surface) в координатах свободных параметров. При этом реальная поверхность ошибки *усредняется* (averaged) по всем возможным примерам, представленным в виде пар "вход-выход". Любое конкретное действие системы с учителем представляется одной точкой на поверхности ошибок. Для повышения производительности системы во времени значение ошибки должно смещаться в сторону

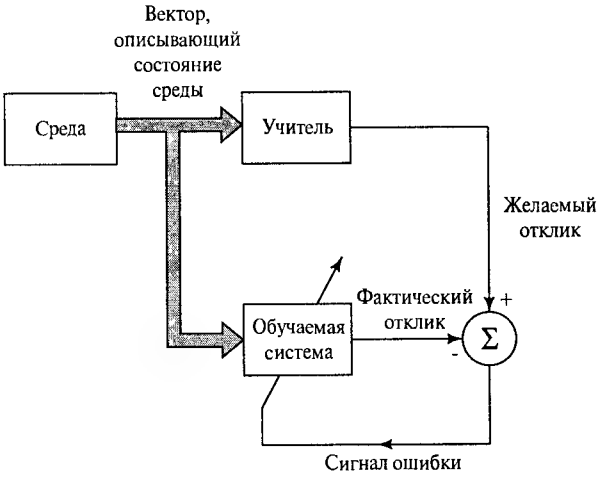


Рис. 2.6. Блочная диаграмма обучения с учителем

минимума на поверхности ошибок. Этот минимум может быть как локальным, так и глобальным. Это можно сделать, если система обладает полезной информацией о *градиенте* поверхности ошибок, соответствующем текущему поведению системы. Градиент поверхности ошибок в любой точке — это вектор, определяющий направление наискорейшего спуска по этой поверхности. В случае обучения с учителем на примерах вычисляется *моментальная оценка* (instantaneous estimate) вектора градиента, в которой входной вектор считается функцией времени. При использовании результатов такой оценки перемещение точки по поверхности ошибок обычно имеет вид "случайного блуждания". Тем не менее при использовании соответствующего алгоритма минимизации функции стоимости, адекватном наборе обучающих примеров в форме "вход-выход" и достаточном времени для обучения системы обучения с учителем способны решать такие задачи, как классификация образов и аппроксимация функций.

2.9. Обучение без учителя

Описанный выше процесс обучения происходит под управлением учителя. Альтернативная парадигма *обучения без учителя* (learning without a teacher) самим названием подчеркивает отсутствие руководителя, контролирующего процесс настройки весовых коэффициентов. При использовании такого подхода не существует маркированных примеров, по которым проводится обучение сети. В этой альтернативной парадигме можно выделить два метода.

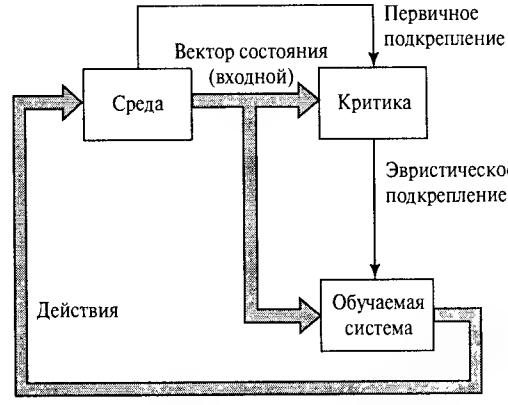


Рис. 2.7. Блочная диаграмма обучения с подкреплением

Обучение с подкреплением, или нейродинамическое программирование

В *обучении с подкреплением*[8] (reinforcement learning) формирование отображения входных сигналов в выходные выполняется в процессе взаимодействия с внешней средой с целью минимизации скалярного индекса производительности. На рис. 2.7 показана блочная диаграмма одной из форм системы обучения с подкреплением, включающей блок "критики", который преобразовывает *первичный сигнал подкрепления* (primary reinforcement signal), полученный из внешней среды, в сигнал более высокого качества, называемый *эвристическим сигналом подкрепления* (heuristic reinforcement signal). Оба этих сигнала являются скалярными [100].

Такая система предполагает *обучение с отложенным подкреплением* (delayed reinforcement). Это значит, что система получает из внешней среды последовательность сигналов возбуждения (т.е. векторов состояния), которые приводят к генерации эвристического сигнала подкрепления. Целью обучения является минимизация *функции стоимости* перехода, определенной как математическое ожидание кумулятивной стоимости *действий*, предпринятых в течение нескольких шагов, а не просто текущей стоимости. Может оказаться, что некоторые предпринятые ранее в данной последо-

[8] Термин "обучение с подкреплением" был введен Минским в его ранних работах, посвященных искусственному интеллекту [742], и, независимо от него, в работе по теории управления [1109]. Тем не менее основная идея "подкрепления" берет свое начало в экспериментальных работах, посвященных обучению животных с точки зрения психологии [415]. В этом контексте следует вспомнить классический *закон влияния* (law of effect) [1052], который сводится к следующему.

"Среди нескольких различных откликов на одну и ту же ситуацию при прочих равных условиях те реакции, которые сопровождались поощрением животного, будут более тесно связаны с этой ситуацией и, следовательно, при ее повторении будут воспроизведены с наибольшей вероятностью. Реакции, которые связаны с дискомфортом, утрачивают свою связь с данной ситуацией и вряд ли будут возобновлены в ответ на ту же ситуацию. Чем выше уровень удовлетворения или дискомфорта, тем значительнее будет усиление или ослабление связи".

Несмотря на то что этот закон нельзя назвать полной моделью биологического поведения, его простота и общий смысл сделали его важным правилом обучения в классическом подходе к обучению с подкреплением.

вательности действия были определяющими в формировании общего поведения всей системы. Функция *обучаемой машины* (learning machine), составляющая второй компонент системы, определяет эти действия и формирует на их основе сигнал обратной связи, направляемый во внешнюю среду.

Практическая реализация обучения с отложенным подкреплением осложнена по двум причинам.

- Не существует учителя, формирующего желаемый отклик на каждом шаге процесса обучения.
- Наличие задержки при формировании первичного сигнала подкрепления требует решения *временной задачи присваивания коэффициентов доверия* (temporal credit assignment). Это значит, что обучаемая машина должна быть способна присваивать коэффициенты доверия и недоверия действиям, выполненным на всех шагах, приводящих к конечному результату, в то время как первичный сигнал подкрепления формируется только на основе конечного результата.

Несмотря на эти сложности, системы обучения с отложенным подкреплением являются очень привлекательными. Они составляют базис систем, взаимодействующих с внешней средой, развивая таким образом способность самостоятельного решения возникающих задач на основе лишь собственных результатов взаимодействия со средой.

Обучение с подкреплением тесно связано с *динамическим программированием* (dynamic programming) — методологией, созданной Беллманом в 1957 году в контексте теории оптимального управления [118]. Динамическое программирование реализует математический формализм последовательного принятия решений. Перемещая обучение с подкреплением в предметную область динамического программирования, можно взять на вооружение все результаты последнего. Это было продемонстрировано в [126]. Введение в проблему динамического программирования и описание его взаимосвязи с обучением с подкреплением будет представлено в главе 12.

Обучение без учителя

Обучение без учителя (unsupervised) (или *обучение на основе самоорганизации* (self-organized)) осуществляется без вмешательства внешнего учителя, или корректора, контролирующего процесс обучения (рис. 2.8). Существует лишь *независимая от задачи мера качества* (task-independent measure) представления, которому должна научиться нейронная сеть, и свободные параметры сети оптимизируются по отношению к этой мере. После обучения сети на статистические закономерности входного сигнала она способна формировать внутреннее представление кодируемых признаков входных данных и, таким образом, автоматически создавать новые классы [112].

Рис. 2.8. Блочная диаграмма обучения без учителя

Для обучения без учителя можно воспользоваться правилом конкурентного обучения. Например, можно использовать нейронную сеть, состоящую из двух слоев — входного и выходного. Входной слой получает доступные данные. Выходной слой состоит из нейронов, конкурирующих друг с другом за право отклика на признаки, содержащиеся во входных данных. В простейшем случае нейронная сеть действует по принципу "победитель получает все". Как было показано в разделе 2.5, при такой стратегии нейрон с наибольшим суммарным входным сигналом "побеждает" в соревновании и переходит в активное состояние. При этом все остальные нейроны отключаются.

Различные алгоритмы обучения без учителя описываются в главах 8 и 11.

2.10. Задачи обучения

В предыдущих разделах описывались различные алгоритмы и парадигмы обучения. В данном разделе рассматривается ряд задач обучения. Выбор конкретного алгоритма обучения зависит от задач, решению которых следует обучить нейронную сеть. В этом контексте можно выделить шесть основных задач, для решения которых в том или ином виде применяются нейронные сети.

Ассоциативная память

Ассоциативная память (associative memory) представляет собой распределенную память, которая обучается на основе ассоциаций, подобно мозгу живых существ. Ассоциативность считается основной характерной чертой человеческого мозга со времен Аристотеля, вследствие чего все модели познания в качестве одной из основных операций в той или иной форме используют ассоциативную память [46].

Существуют два типа ассоциативной памяти: *автоассоциативная* (autoassociation) и *гетероассоциативная* (heteroassociation). При решении задачи автоассоциативной памяти в нейронной сети *запоминаются* передаваемые ей образы (векторы). Затем в эту сеть последовательно подаются неполные описания или зашумленные представления хранимых в памяти исходных образов, и ставится задача *распознавания* конкретного образа. Гетероассоциативная память отличается от автоассоциативной тем, что произвольному набору входных образов ставится в соответствие другой произвольный набор выходных сигналов. Для настройки нейронных сетей, предна-

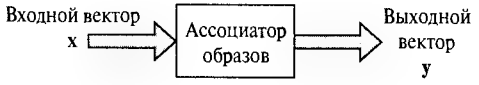


Рис. 2.9. Диаграмма "вход-выход" для сети ассоциации образов

значенных для решения задач автоассоциативной памяти, используется обучение без учителя, а в сетях гетероассоциативной памяти — обучение с учителем.

Пусть $\mathbf{x}_k$ — *ключевой образ* (key pattern) (вектор), применяемый для решения задачи ассоциативной памяти, а $\mathbf{y}_k$ — *запомненный образ* (memorized pattern) (вектор). Отношение ассоциации образов, реализуемое такой сетью, можно описать следующим образом:

$$\mathbf{x}_k \rightarrow \mathbf{y}_k, k = 1, 2, \ldots, q, \tag{2.18}$$

где q — количество хранимых в сети образов. Ключевой образ $\mathbf{x}_k$ выступает в роли стимула, который не только определяет местоположение запомненного образа $\mathbf{y}_k$, но и содержит ключ для его извлечения.

В автоассоциативной памяти $\mathbf{y}_k = \mathbf{x}_k$. Это значит, что пространства входных и выходных данных сети должны иметь одинаковую размерность. В гетероассоциативной памяти $\mathbf{y}_k \neq \mathbf{x}_k$. Это значит, что размерность пространства выходных векторов может отличаться от размерности пространства входных (но может и совпадать с ней).

В работе ассоциативной памяти можно выделить две фазы.

- *Фаза запоминания* (storage phase), которая соответствует процессу обучения сети в соответствии с формулой (2.18).
- *Фаза восстановления* (recall phase), соответствующая извлечению запомненного образа в ответ на представление в сеть зашумленной или искаженной версии ключа.

Пусть стимул (входной сигнал) $\mathbf{x}$ представляет собой зашумленную или искаженную версию ключевого образа $\mathbf{x}_j$. Этот стимул вызывает отклик (выходной сигнал) $\mathbf{y}$ (рис. 2.9). В идеальном случае $\mathbf{y} = \mathbf{y}_j$, где $\mathbf{y}_j$ — запомненный образ, ассоциируемый с ключом $\mathbf{x}_j$. Если при $\mathbf{x} = \mathbf{x}_j$ выход сети $\mathbf{y} \neq \mathbf{y}_j$, то говорят, что ассоциативная память при восстановлении образа сделала *ошибку* (error).

Количество q образов, хранимых в ассоциативной памяти, является непосредственной мерой *емкости памяти* (storage capacity) сети. При построении ассоциативной памяти желательно максимально увеличить ее емкость (информационная емкость ассоциативной памяти измеряется в процентах от общего количества нейронов N, использованных для создания сети) и убедиться, что большая часть запомненных образов восстанавливается корректно.

Распознавание образов

Человеческий мозг хорошо приспособлен для распознавания образов. Мы получаем данные из окружающего мира через сенсоры и способны распознать источник данных. Чаще всего это выполняется мгновенно и без всяких усилий. Например, мы можем узнать знакомое лицо, даже если наш знакомый с момента последней встречи сильно постарел. Человек может узнать знакомый голос по телефону, несмотря на помехи в линии связи. Мы можем по запаху отличить свежее яйцо от тухлого. Человек распознает образы в результате процесса обучения. То же происходит и с нейронными сетями.

Распознавание образов формально определяется как *процесс, в котором получаемый образ/сигнал должен быть отнесен к одному из предопределенных классов (категорий)*. Чтобы нейронная сеть могла решать задачи распознавания образов, ее сначала необходимо обучить, подавая последовательность входных образов наряду с категориями, которым эти образы принадлежат. После обучения сети на вход подается ранее не виденный образ, который принадлежит тому же набору категорий, что и множество образов, использованных при обучении. Благодаря информации, выделенной из данных обучения, сеть сможет отнести представленный образ к конкретному классу. Распознавание образов, выполняемое нейронной сетью, является по своей природе статистическим. При этом образы представляются отдельными точками в многомерном *пространстве решений*. Все пространство решений разделяется на отдельные области, каждая из которых ассоциируется с определенным классом. Границы этих областей как раз и формируются в процессе обучения. Построение этих границ выполняется статистически на основе дисперсии, присущей данным отдельных классов.

В целом машины распознавания образов, созданные на основе нейронных сетей, можно разделить на два типа.

- Система состоит из двух частей: сети *извлечения признаков* (future extraction) (без учителя) и сети *классификации* (classification) (с учителем) (рис. 2.10, *а*). Такой метод соответствует традиционному подходу к статистическому распознаванию образов [269], [322]. В концептуальных терминах образ представляется как набор из m наблюдений, каждое из которых можно рассматривать как точку $\mathbf{x}$ в m-мерном *пространстве наблюдений* (*данных*) (observation (data) space). Извлечение признаков описывается с помощью преобразования, которое переводит точку $\mathbf{x}$ в промежуточную точку $\mathbf{y}$ в q-мерном пространстве признаков, где $q < m$ (рис. 2.10, *б*). Это преобразование можно рассматривать как операцию снижения размерности (т.е. сжатие данных), упрощающую задачу классификации. Сама классификация описывается как преобразование, которое отображает промежуточную точку $\mathbf{y}$ в один из классов r-мерного пространства решений (где r — количество выделяемых классов).

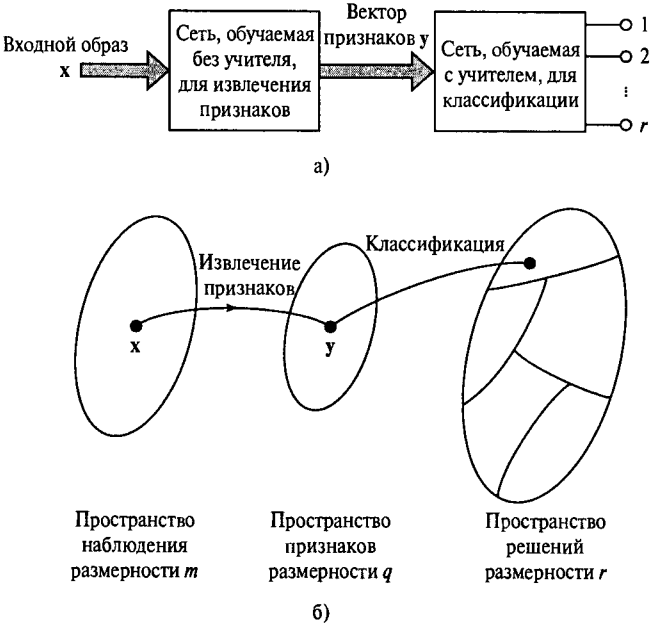


Рис. 2.10. Иллюстрация классического подхода к распознаванию образов

- Система проектируется как единая многослойная сеть прямого распространения, использующая один из алгоритмов обучения с учителем. При этом подходе задача извлечения признаков выполняется вычислительными узлами скрытого слоя сети.

Какой из этих двух подходов применять на практике — зависит от конкретной предметной области.

Аппроксимация функций

Третьей задачей обучения является аппроксимация функций. Рассмотрим нелинейное отображение типа "вход-выход", заданное следующим соотношением:

$$\mathbf{d} = \mathbf{f}(\mathbf{x}), \tag{2.19}$$

где вектор $\mathbf{x}$ — вход, а вектор $\mathbf{d}$ — выход. Векторная функция $\mathbf{f}(\cdot)$ считается неизвестной. Чтобы восполнить пробел в знаниях о функции $\mathbf{f}(\cdot)$, нам предоставляется множество маркированных примеров:

$$\mathrm{T} = \{(\mathbf{x}_i, \mathbf{d}_i)\}_{i=1}^{N}. \tag{2.20}$$

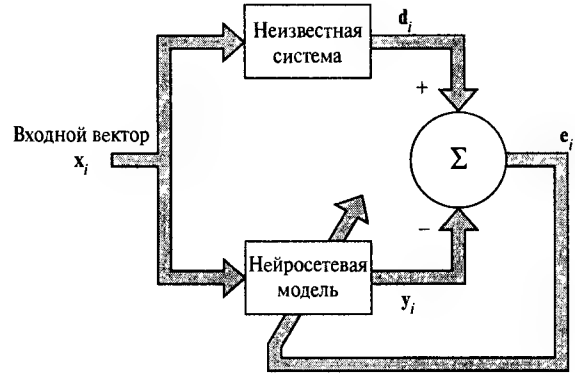


Рис. 2.11. Блочная диаграмма решения задачи идентификации системы

К структуре нейронной сети, аппроксимирующей неизвестную функцию $\mathbf{f}(\cdot)$, предъявляется следующее требование: функция $\mathbf{F}(\cdot)$, описывающая отображение входного сигнала в выходной, должна быть *достаточно близка* к функции $\mathbf{f}(\cdot)$ в смысле Евклидовой нормы на множестве всех входных векторов $\mathbf{x}$, т.е.

$$||\mathbf{F}(\mathbf{x}) - \mathbf{f}(\mathbf{x})|| < \varepsilon \text{ для всех векторов } \mathbf{x}, \tag{2.21}$$

где ε — некоторое малое положительное число. Если количество N элементов обучающего множества достаточно велико и в сети имеется достаточное число свободных параметров, то ошибку аппроксимации ε можно сделать достаточно малой.

Описанная задача аппроксимации является отличным примером задачи для обучения с учителем. Здесь $\mathbf{x}_i$ играет роль входного вектора, а $\mathbf{d}_i$ — роль желаемого отклика. Исходя из этого, задачу обучения с учителем можно свести к задаче аппроксимации.

Способность нейронной сети аппроксимировать неизвестное отображение входного пространства в выходное можно использовать для решения следующих задач.

- *Идентификация систем* (system identification). Пусть формула (2.19) описывает соотношение между входом и выходом в неизвестной системе с несколькими входами и выходами (MIMO-системе) без памяти. Термин "без памяти" подразумевает инвариантность системы во времени. Тогда множество маркированных примеров (2.20) можно использовать для обучения нейронной сети, представляющей модель этой системы. Пусть $\mathbf{y}_i$ — выход нейронной сети, соответствующий входному вектору $\mathbf{x}_i$. Разность между желаемым откликом $\mathbf{d}_i$ и выходом сети $\mathbf{y}_i$ составляет вектор сигнала ошибки $\mathbf{e}_i$ (рис. 2.11), используемый для корректировки свободных параметров сети с целью минимизации среднеквадратической ошибки — суммы квадратов разностей между выходами неизвестной системы и нейронной сети в статистическом смысле (т.е. вычисляемой на множестве всех примеров).

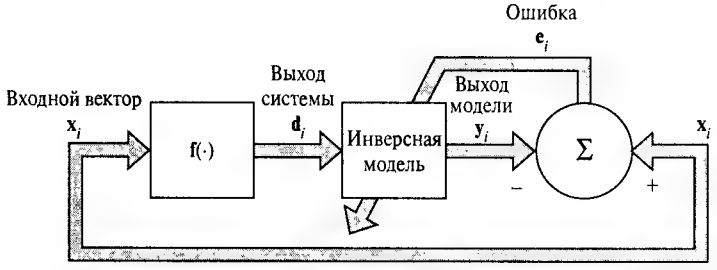


Рис. 2.12. Блочная диаграмма моделирования инверсной системы

- *Инверсные системы* (inverse system). Предположим, что существует некая система MIMO без памяти, для которой преобразование входного пространства в выходное описывается соотношением (2.19). Требуется построить инверсную систему, которая в ответ на вектор $\mathbf{d}$ генерирует отклик в виде вектора $\mathbf{x}$. Инверсную систему можно описать следующим образом:

$$\mathbf{x} = \mathbf{f}^{-1}(\mathbf{d}), \tag{2.22}$$

где вектор-функция $\mathbf{f}^{-1}(\cdot)$ является инверсной к функции $\mathbf{f}(\cdot)$. Обратим внимание, что функция $\mathbf{f}^{-1}(\cdot)$ не является обратной к функции $\mathbf{f}(\cdot)$. Здесь верхний индекс -1 используется только как индикатор инверсии. Во многих ситуациях на практике функция $\mathbf{f}(\cdot)$ может быть достаточно сложной, что делает практически невозможным формальный вывод обратной функции $\mathbf{f}^{-1}(\cdot)$. Однако на основе множества маркированных примеров (2.20) можно построить нейронную сеть, аппроксимирующую обратную функцию $\mathbf{f}^{-1}(\cdot)$ с помощью схемы, изображенной на рис. 2.12. В данной ситуации роли векторов $\mathbf{x}_i$ и $\mathbf{d}_i$ меняются: вектор $\mathbf{d}_i$ используется как входной сигнал, а вектор $\mathbf{x}_i$ — как желаемый отклик. Пусть вектор сигнала ошибки определяется как разность между вектором $\mathbf{x}_i$ и выходом нейронной сети $\mathbf{y}_i$, полученным в ответ на возмущение $\mathbf{d}_i$. Как и в задаче идентификации системы, вектор сигнала ошибки используется для корректировки свободных параметров нейронной сети с целью минимизации суммы квадратов разностей между выходами неизвестной инверсной системы и нейронной сети в статистическом смысле (т.е. вычисляемой на всем множестве примеров обучения).

Управление

Управление *предприятием* (plant) является еще одной задачей обучения, которую можно решать с использованием нейронной сети. Здесь под термином "предприятие" понимается процесс или критическая часть системы, подлежащие управлению. Ассоциация между управлением и обучением вряд ли кого-нибудь удивит, так как человеческий мозг является компьютером (обработчиком информации), выходами ко-

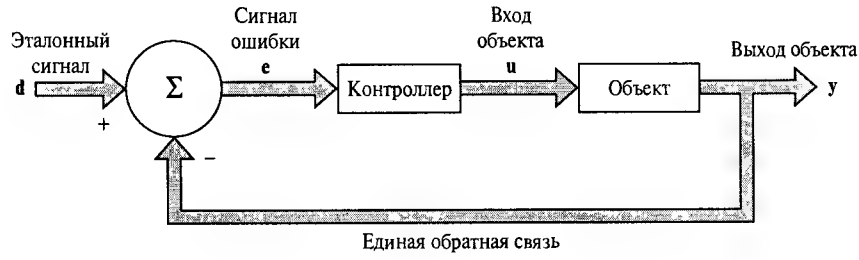


Рис. 2.13. Блочная диаграмма системы управления с обратной связью

торого, как системы в целом, являются *действия*. В контексте задач управления сам мозг является живым доказательством возможности создания обобщенной системы управления, использующей преимущества параллельных распределенных вычислений и одновременно управляющей тысячами функциональных механизмов. Такая система является нелинейной, может обрабатывать шумы и оптимизировать свою работу в ракурсе долгосрочного планирования [1125].

Рассмотрим *систему управления с обратной связью* (feedback control system), показанную на рис. 2.13. В этой системе используется единственная обратная связь, охватывающая весь объект управления (т.е. выход объекта связан с его входом)[9]. Таким образом, выход объекта управления **y** вычитается из *эталонного сигнала* (reference signal) **d**, принимаемого из внешнего источника. Полученный таким образом сигнал ошибки **e** подается на *нейроконтроллер* (neurocontroller) для настройки свободных параметров. Основной задачей контроллера является поддержание такого входного вектора объекта, для которого выходной сигнал **y** соответствует эталонному значению **d**. Другими словами, в задачу контроллера входит инвертирование отображения вход-выход объекта управления.

Заметим, что на рис. 2.13 сигнал ошибки **e** распространяется через нейроконтроллер, прежде чем достигнет объекта управления. Следовательно, для настройки свободных параметров объекта управления в соответствии с алгоритмом обучения на основе коррекции ошибок необходимо знать матрицу Якобиана:

$$\mathbf{J} = \left\{ \frac{\partial y_k}{\partial u_j} \right\}, \tag{2.23}$$

где y_k — элемент выходного сигнала **y** объекта управления; u_j — элемент вектора входов объекта **u**. К сожалению, частные производные $\partial y_k / \partial u_j$ для различных k и j

[9] Выходной сигнал объекта обычно является некоторой физической переменной. Для управления объектом необходимо знать значение этой переменной, а значит, необходимо измерять выходной сигнал объекта. Система, используемая для измерения физических величин, называется датчиком, или *сенсором*. Таким образом, чтобы быть предельно точными, блочная диаграмма на рис. 2.13 должна содержать сенсор в цепи обратной связи. Мы опустили использование сенсора, считая, что передаточная функция сенсора тождественно равна единичной.

зависят от рабочей точки объекта управления и, следовательно, не известны. Для их оценки можно использовать два следующих подхода.

- *Непрямое обучение* (indirect learning). С использованием текущих измерений входа и выхода объекта управления строится модель нейронной сети, воспроизводящая эту зависимость. Эта модель, в свою очередь, используется для оценки матрицы Якобиана $\mathbf{J}$. Частные производные, составляющие эту матрицу, впоследствии используются в алгоритме обучения на основе коррекции ошибки для настройки свободных параметров нейроконтроллера [782], [1037], [1145].
- *Прямое обучение* (direct learning). Знаки частных производных $\partial y_k / \partial u_j$ в общем случае известны и обычно остаются постоянными в некотором динамическом диапазоне значений объекта управления. Это наводит на мысль, что частные производные можно аппроксимировать по их знакам. Абсолютные значения частных производных задаются распределенным представлением в свободных параметрах нейроконтроллера [923], [942]. Таким образом, свободные параметры нейроконтроллера можно настроить непосредственно с помощью объекта управления.

Фильтрация

Под термином *фильтр* (filter) обычно понимают устройство или алгоритм, используемые для извлечения полезной информации из набора зашумленных данных. Шум может возникать по многим причинам. Например, данные могут быть измерены с помехами, или возмущения могут возникнуть при передаче информационного сигнала через зашумленные линии связи. Кроме того, на полезный сигнал может быть наложен другой сигнал, поступающий из окружающей среды. Фильтры можно применять для решения трех основных задач обработки информации.

1. *Фильтрация* (filtering), т.е. извлечение полезной информации в дискретные моменты времени n из данных, измеренных вплоть до момента времени n включительно.
2. *Сглаживание* (smoothing). Эта задача отличается от фильтрации тем, что информация о полезном сигнале в момент времени n не требуется, поэтому для извлечения этой информации можно использовать данные, полученные позже. Это значит, что в сглаживании при формировании результата присутствует *запаздывание* (delay). Так как при сглаживании можно использовать данные, полученные не только до момента времени n, но и после него, этот процесс в статистическом смысле является более точным, чем фильтрация.
3. *Прогнозирование* (prediction). Целью этого процесса является получение прогноза относительно состояния объекта управления в некоторый момент времени $n + n_0$, где $n_0 > 0$, на основе данных, полученных до момента n (включительно).

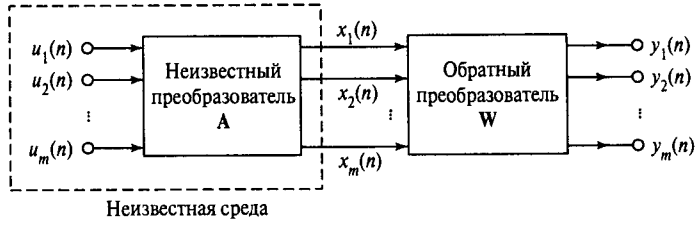


Рис. 2.14. Блочная диаграмма слепого разделения источников

Задачей фильтрации, с которой знакомы практически все, является *задача распознавания голоса собеседника на шумной вечеринке*[10] (cocktail party problem). Мы обладаем способностью сконцентрироваться на голосе собеседника, несмотря на разноголосый шум, доносящийся от групп посетителей вечеринки, слова которых мы не различаем. Вероятно, для решения этой задачи используется некоторая форма предсознательного и преаттентивного анализа [1090]. В контексте (искусственной) нейронной сети аналогичная задача фильтрации возникает при *слепом разделении сигнала* (blind signal separation) [37], [116], [205]. Чтобы сформулировать задачу слепого разделения сигнала, представим себе неизвестный источник независимых сигналов $\{s_i(n)\}_{i=1}^{m}$. Эти сигналы линейно смешиваются неизвестным сенсором, в результате чего получается вектор наблюдения размерности $m \times 1$ (рис. 2.14):

$$\mathbf{x}(n) = \mathbf{A}\mathbf{u}(n), \tag{2.24}$$

где

$$\mathbf{u}(n) = [u_1(n), u_2(n), \ldots, u_m(n)]^T, \tag{2.25}$$

$$\mathbf{x}(n) = [x_1(n), x_2(n), \ldots, x_m(n)]^T, \tag{2.26}$$

где $\mathbf{A}$ — неизвестная несингулярная *матрица смешивания* размерности $m \times m$. Вектор наблюдения $\mathbf{x}(n)$ известен. Требуется восстановить исходный сигнал $u_1(n), u_2(n), \ldots, u_m(n)$ методом обучения без учителя.

---

[10] Так называемый "феномен вечеринки" (cocktail party phenomenon) связан с уникальной способностью человека избирательно отфильтровывать нужные ему источники звукового сигнала в зашумленной среде [188], [189]. Эта способность обеспечивается комбинацией трех процессов, выполняемых слуховым аппаратом.

Сегментация. Входной звуковой сигнал сегментируется на отдельные каналы, несущие значимую для слушателя информацию об окружающей среде. Среди эвристик, используемых слушателем при сегментации сигнала, самой важной является пространственное расположение местоположения объекта (spatial location) [753].

Внимание. Это способность слушателя фокусировать внимание на одном из каналов при одновременной блокировке внимания к другим каналам [188].

Переключение. Способность переключать внимание с одного канала на другой, что обычно обеспечивается за счет открытия-закрытия шлюзов каналов входного звукового сигнала по принципу "сверху вниз" [1165].

Из этого можно сделать вывод, что обработка звукового сигнала носит *пространственно-временной* характер.

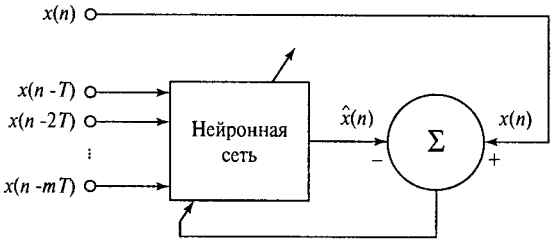


Рис. 2.15. Блочная диаграмма нелинейного прогнозирования

Вернемся к задаче прогнозирования, в которой требовалось предсказать текущее значение процесса $x(n)$ по предшествующей информации об этом процессе, снятой в дискретные моменты времени и представленной значениями $x(n-T), x(n-2T), \ldots, x(n-mT)$, где T — периодичность снятия сигнала, а m — порядок прогнозирования. Задача прогнозирования может быть решена с помощью обучения на основе коррекций ошибок без учителя, так как обучающие примеры получаются непосредственно от самого процесса (рис. 2.15). При этом $x(n)$ выступает в роли желаемого отклика. Обозначим $\hat{x}(n)$ результат прогнозирования на один шаг вперед, сгенерированный нейронной сетью в момент времени n. Сигнал ошибки определяется как разность между $x(n)$ и $\hat{x}(n)$. Он используется для настройки свободных параметров нейронной сети. При этих допущениях прогнозирование можно рассматривать как форму *построения модели* (model building) в том смысле, что чем меньше ошибка прогнозирования в статистическом смысле, тем лучше нейронная сеть будет работать в качестве модели имитируемого физического процесса, обеспечивающего генерацию данных. Если этот процесс является *нелинейным*, нейронная сеть является мощным средством решения задачи прогнозирования, так как нелинейная обработка сигнала предполагается самой архитектурой нейронной сети. Единственное исключение составляет выходной слой нейронов: если динамический диапазон временного ряда $\{x(n)\}$ не известен, в выходном слое целесообразно использовать линейные обрабатывающие элементы.

Формирование диаграммы направленности

Формирование диаграммы направленности является *пространственным* (spatial) случаем фильтрации и используется для разделения пространственных признаков полезного сигнала и шумов. Для формирования диаграммы направленности используется *специальное устройство*.

Задачу формирования диаграммы направленности целесообразно решать с использованием нейронных сетей, созданных на основе знаний о строении слухового аппарата человека [151] и решении задач эхо-локации в корковых слоях системы слуха летучей мыши [994], [1026]. Система эхо-локации летучей мыши отфильтровывает влияние окружающей среды. Мышь испускает короткие ультразвуковые сигналы с ча-

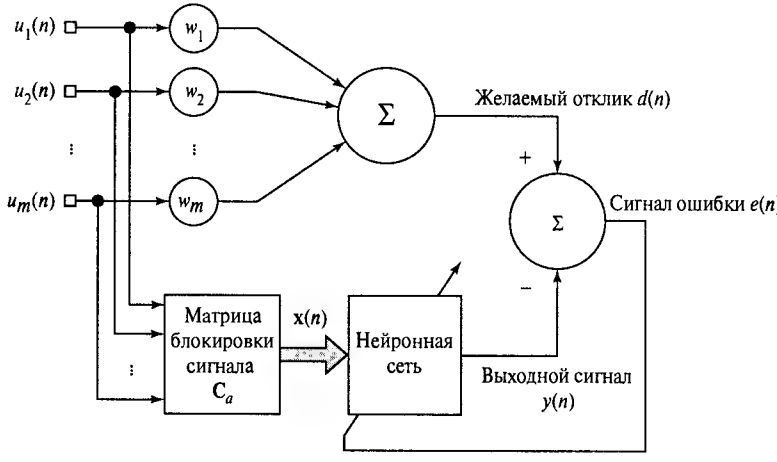


Рис. 2.16. Блочная диаграмма обобщенной системы подавления боковых лепестков

стотной модуляцией, а затем улавливает отраженные сигналы с помощью слухового аппарата (пары ушей), фокусируя внимание на потенциальной добыче (насекомых). Уши летучей мыши выполняют некоторый вид пространственной фильтрации (а точнее, *интерферометрии*), результаты которой используются слуховой системой для извлечения сигнала, содержащего интересующий *класс объектов* (attentional selectivity).

Построение диаграммы направленности, как правило, выполняется в системах радаров, где задача сводится к отслеживанию траектории цели в условиях помех и интерференции (т.е. наложения сигналов друг на друга). Эта задача усложняется двумя следующими факторами.

- Полезный сигнал формируется в неизвестном направлении.
- Не существует *априорной* информации о наложении сигналов в результате интерференции.

Одним из способов решения проблем такого рода является подавление боковых лепестков. Блочная диаграмма устройства, позволяющего решать такую задачу, показана на рис. 2.16. Эта система состоит из следующих компонентов [386], [434], [1083].

- *Массив антенных элементов* (array of antenna elements), который снимает сигналы в дискретных точках пространства.
- *Линейный сумматор* (linear combiner), определяемый множеством фиксированных весов $\{w_i\}_{i=1}^{m}$, на выходе которого формируется желаемый отклик. Работа этого линейного сумматора подобна пространственному фильтру, характеризуемому диаграммой излучения (т.е. графиком амплитуды выходного сигнала антенны в зависимости от угла поступающего сигнала в полярных координатах). Главный лепе-

сток этой диаграммы излучения определяет направление, для которого обобщенная система подавления боковых лепестков выдает неискаженный отклик. Выход этого линейного сумматора, обозначенный $d(n)$, как раз и обеспечивает желаемый отклик обобщенной системы подавления боковых лепестков.

- *Матрица блокировки сигнала* (signal-blocking matrix) $\mathbf{C}_a$, функцией которой является блокировка влияния сторонних сигналов, которые просачиваются через боковые лепестки диаграммы излучения пространственного фильтра, представляющего линейный сумматор.
- *Нейронная сеть* с настраиваемыми параметрами, предназначенная для адаптации к статистическим вариациям сторонних сигналов.

Настройка свободных параметров нейронной сети выполняется с помощью алгоритма обучения на основе коррекции ошибки, определяемой как разность между выходом линейного сумматора $d(n)$ и фактическим выходным сигналом $y(n)$ нейронной сети. Таким образом, обобщенная система подавления боковых лепестков GSLC (generalized sidelobe canceller) работает под управлением линейного сумматора, выполняющего роль учителя. Как и при обычном обучении с учителем, линейный сумматор находится вне контура обратной связи нейронной сети. Обобщенная система подавления боковых лепестков, для обучения которой используется нейронная сеть, называется *нейросетевой системой подавления боковых лепестков* (neurobeamformer). Этот класс обучаемых машин получил общее название нейрокомпьютеров, предназначенных для настройки внимания [447].

Разнообразие представленных здесь шести задач обучения еще раз свидетельствует в пользу универсальности нейронных сетей как систем обработки информации. В фундаментальном смысле все эти задачи являются проблемами обучения построению отображений на основе примеров (иногда зашумленных). При отсутствии априорных знаний все эти задачи являются *плохо обусловленными* (ill posed), т.е. их возможные решения определяются неоднозначно. Одним из способов обеспечения *хорошей обусловленности* (well pose) решений является применение теории регуляризации, которая описывается в главе 5.

2.11. Память

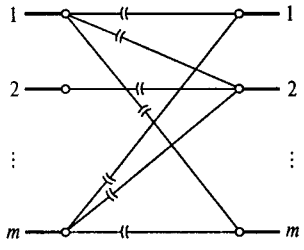
Обсуждение задач обучения, особенно задачи ассоциативной памяти, заставляет задуматься о самой *памяти* (memory). В контексте нейробиологии под памятью понимается относительно продолжительная во времени деформация структуры нейронов, вызванная влиянием на организм внешней среды [1051]. Без такой деформации память не существует. Чтобы память была полезной, она должна быть доступной для нервной системы. Только тогда она может влиять на будущее поведение организма.

Однако для этого в памяти предварительно должны быть накоплены определенные модели поведения. Это накопление осуществляется с помощью *процесса обучения* (learning process). Память и обучение тесно взаимосвязаны. При изучении некоторого образа он сохраняется в структуре мозга, откуда может быть впоследствии извлечен в случае необходимости. Формально память можно разделить на кратковременную и долговременную, в зависимости от сроков возможного хранения информации [66]. *Кратковременная память* является отображением текущего состояния окружающей среды. Каждое "новое" состояние среды, отличное от образа, содержащегося в кратковременной памяти, приводит к обновлению данных в памяти. С другой стороны, в *долговременной памяти* хранятся знания, предназначенные для продолжительного (или даже постоянного) использования.

В этом разделе рассматриваются вопросы ассоциативной памяти, которая характеризуется следующими особенностями.

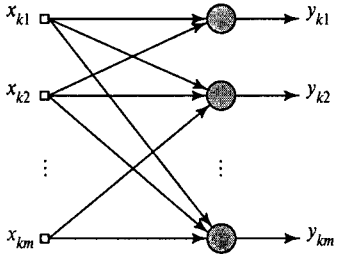
- Эта память является распределенной.
- Стимулы (ключи) и отклики (хранимые образы) ассоциативной памяти представляют собой векторы данных.
- Информация запоминается с помощью формирования пространственных образов нейросетевой активности на большом количестве нейронов.
- Информация, содержащаяся в стимуле, определяет не только место ее запоминания, но и адрес для ее извлечения.
- Несмотря на то что нейроны не являются надежными вычислительными элементами и работают в условиях шума, память обладает высокой устойчивостью к помехам и искажению данных.
- Между отдельными образами, хранимыми в памяти, могут существовать внутренние взаимосвязи. (В противном случае размер памяти должен быть невероятно велик, чтобы вместить все отдельные изолированные образы.) Отсюда и проистекает вероятность *ошибок* при извлечении информации из памяти.

В *распределенной памяти* (distributed memory) главный интерес представляет одновременное или почти параллельное функционирование множества различных нейронов при обработке внутренних или внешних стимулов. Нейронная активность формирует в памяти пространственные образы, содержащие информацию о стимулах. Таким образом, память выполняет распределенное отображение образов в пространстве входных сигналов в другие образы выходного пространства. Некоторые важные свойства отображения распределенной памяти можно проиллюстрировать на примере идеализированной нейросети, состоящей из двух слоев нейронов. На рис. 2.17, *а* показана сеть, которую можно рассматривать как *модельный компонент нервной системы* (model component of nervous system) [210], [953]. Все нейроны входного слоя соединены со всеми нейронами выходного. В реальных системах синаптические связи между



Входной слой нейронов | Синаптические соединения | Выходной слой нейронов

а) Компонент модели ассоциативной памяти нервной системы



Входной слой узлов источника | Выходной слой нейронов

б) Модель ассоциативной памяти с искусственными нейронами

Рис. 2.17. Модели ассоциативной памяти

нейронами являются сложными и избыточными. В модели, показанной на рис. 2.17, *а*, для представления общего результата от всех синаптических контактов между дендритами нейронов входного слоя и ответвлениями аксонов нейронов выходного слоя использовались идеальные соединения. Уровень активности нейрона входного слоя может оказывать влияние на степень активности любого нейрона выходного слоя.

Аналогичная ситуация для искусственной нейронной сети показана на рис. 2.17, *б*. В данном случае узлы источника из входного слоя и нейроны выходного слоя работают как вычислительные элементы. Синаптические веса интегрированы в нейроны выходного слоя. Связи между двумя слоями сети представляют собой простые "проволочные" соединения.

В нижеследующих математических выкладках нейронные сети, изображенные на рис. 2.17, считаются *линейными*. В результате этого предположения все нейроны рассматриваются как линейные сумматоры, что и показано на графе передачи сигнала (рис. 2.18). Для дальнейшего анализа предположим, что во входной слой передается образ $\mathbf{x}_k$, при этом в выходном слое возникает образ $\mathbf{y}_k$. Рассмотрим вопрос обуче-

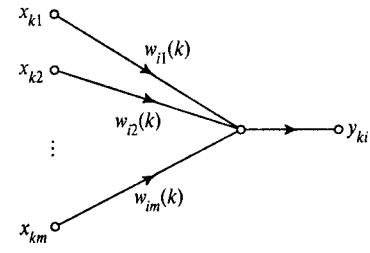


Рис. 2.18. Граф передачи сигнала для i-го линейного нейрона

ния на основе ассоциации между образами $\mathbf{x}_k$ и $\mathbf{y}_k$. Образы $\mathbf{x}_k$ и $\mathbf{y}_k$ представлены векторами, развернутая форма которых имеет следующий вид:

$$\mathbf{x}_k(n) = [x_{k1}(n), x_{k2}(n), \ldots, x_{km}(n)]^T,$$
$$\mathbf{y}_k(n) = [y_{k1}(n), y_{k2}(n), \ldots, y_{km}(n)]^T.$$

Для удобства представления предположим, что размерности пространств входных и выходных векторов совпадают и равны m, т.е. размерности векторов $\mathbf{x}_k$ и $\mathbf{y}_k$ одинаковы. Исходя из этого, значение m будем называть *размерностью сети* (network dimensionality), или просто *размерностью*. Заметим, что значение m равно количеству узлов источника во входном слое и числу вычислительных нейронов выходного слоя. В реальных нейронных сетях размерность m может быть достаточно большой.

Элементы векторов $\mathbf{x}_k$ и $\mathbf{y}_k$ могут принимать как положительные, так и отрицательные значения. В искусственных нейронных сетях такое допущение является естественным. Такая ситуация характерна и для нервной системы, если рассматривать переменную, равную разности между фактическим уровнем активности (т.е. степенью возбуждения нейрона) и произвольным ненулевым уровнем активности.

Учитывая линейность сети, показанной на рис. 2.17, ассоциацию между ключевым вектором $\mathbf{x}_k$ и запомненным вектором $\mathbf{y}_k$ можно представить в матричном виде:

$$\mathbf{y}_k = \mathbf{W}(k)\mathbf{x}_k \text{ , } k=1, 2, \ldots, q, \tag{2.27}$$

где $\mathbf{W}(k)$ — матрица весов, определяемая парами "вход-выход" $(\mathbf{x}_k, \mathbf{y}_k)$.

Чтобы детально описать матрицу весовых коэффициентов $\mathbf{W}(k)$, обратимся к рис. 2.18, на котором представлена схема i-го нейрона выходного слоя. Выход y_{ik} этого нейрона вычисляется как взвешенная сумма элементов ключевого образа $\mathbf{x}_k$ по следующей формуле:

$$y_{ki} = \sum_{j=1}^{m} w_{ij}(k)\, x_{kj}, \quad i = 1, 2, \ldots, m, \tag{2.28}$$

где $w_{ij}(k), j = 1, 2, \ldots, m$ — синаптические веса нейрона i, соответствующие k-й паре ассоциированных образов. Используя матричное представление, элемент y_{ki} можно записать в эквивалентном виде:

$$y_{ki} = [w_{i1}(k),\ w_{i2}(k),\ \ldots,\ w_{im}(k)] \begin{bmatrix} x_{k1} \\ x_{k2} \\ \ldots \\ x_{km} \end{bmatrix},\ i = 1,\ 2,\ \ldots,\ m. \tag{2.29}$$

Вектор-столбец в правой части равенства (2.29) представляет собой ключевой вектор $\mathbf{x}_k$. Подставляя выражение (2.29) в определение запоминаемого вектора $\mathbf{y}_k$ размерности $m \times 1$, получим:

$$\begin{bmatrix} y_{k1} \\ y_{k2} \\ \ldots \\ y_{km} \end{bmatrix} = \begin{bmatrix} w_{11}(k) & w_{12}(k) & \ldots & w_{1m}(k) \\ w_{21}(k) & w_{22}(k) & \ldots & w_{2m}(k) \\ \ldots & \ldots & \ldots\ldots & \\ w_{m1}(k) & w_{m2}(k) & \ldots & w_{mm}(k) \end{bmatrix} \begin{bmatrix} x_{k1} \\ x_{k2} \\ \ldots \\ x_{km} \end{bmatrix}. \tag{2.30}$$

Соотношение (2.30) описывает матричное преобразование (или отображение) (2.27) в развернутом виде. В частности, матрица $\mathbf{W}(k)$ размерности $m \times m$ определяется в виде

$$\mathbf{W}(k) = \begin{bmatrix} w_{11}(k) & w_{12}(k) & \ldots & w_{1m}(k) \\ w_{21}(k) & w_{22}(k) & \ldots & w_{2m}(k) \\ \ldots & \ldots & \ldots\ldots & \\ w_{m1}(k) & w_{m2}(k) & \ldots & w_{mm}(k) \end{bmatrix}. \tag{2.31}$$

Отдельные представления q пар ассоциированных образов $\mathbf{x}_k \rightarrow \mathbf{y}_k, k = 1, 2, \ldots, q$ формируют значения элементов отдельных матриц $\mathbf{W}(1)$, $\mathbf{W}(2)$, $\ldots$, $\mathbf{W}(q)$. Учитывая тот факт, что эта ассоциация образов представляется матрицей весов $\mathbf{W}(k)$, *матрицу памяти* (memory matrix) размерности $m \times m$ можно определить как сумму матриц весовых коэффициентов всего набора ассоциаций:

$$\mathbf{M} = \sum_{k=1}^{q} \mathbf{W}(k). \tag{2.32}$$

Матрица $\mathbf{M}$ определяет связи между входным и выходным слоями ассоциативной памяти. Она представляет *опыт* (experience), накопленный в результате подачи q образов, представленных в виде пар "вход-выход". Другими словами, в матрице $\mathbf{M}$ содержатся данные обо всех парах "вход-выход", представленных для записи в память.

Описание матрицы памяти (2.32) можно представить в рекурсивной форме

$$\mathbf{M}_k = \mathbf{M}_{k-1} + \mathbf{W}(k), k = 1, 2, \ldots, q, \tag{2.33}$$

где исходная матрица $\mathbf{M}_0$ является нулевой (т.е. все синаптические веса памяти изначально равны нулю), а окончательная матрица $\mathbf{M}_q$ совпадает с матрицей $\mathbf{M}$, определенной выражением (2.32). В соответствии с рекурсивной формулой (2.33) под обозначением $\mathbf{M}_{k-1}$ понимается матрица, полученная на $k+1$ шаге ассоциации, а под $\mathbf{M}_k$ понимается обновленная матрица, полученная в результате добавления приращения $\mathbf{W}(k)$, полученного на основе k-й пары векторов. При этом следует заметить, что при добавлении к матрице $\mathbf{M}_{k-1}$ приращение $\mathbf{W}(k)$ теряет свою идентичность в сумме комбинаций, формирующих матрицу $\mathbf{M}$. Заметим, что при увеличении числа q хранимых образов влияние каждого из них на состояние памяти ослабляется.

Память в виде матрицы корреляции

Предположим, что ассоциативная память (см. рис. 2.17, *б*) была обучена на парах векторов входного и выходного сигналов $\mathbf{x}_k \rightarrow \mathbf{y}_k$, в результате чего была вычислена матрица памяти $\mathbf{M}$. Для оценки матрицы $\mathbf{M}$ введем обозначение $\hat{\mathbf{M}}$, которое в терминах запоминаемых образов описывается выражением [50], [210]

$$\hat{\mathbf{M}} = \sum_{k=1}^{q} \mathbf{y}_k \mathbf{x}_k^T. \tag{2.34}$$

Под обозначением $\mathbf{y}_k \mathbf{x}_k^T$ понимается *внешнее (матричное) произведение* (outer product) ключевого $\mathbf{x}_k$ и запомненного $\mathbf{y}_k$ образов. Это произведение является оценкой матрицы весов $\mathbf{W}(k)$, которая отображает выходной вектор $\mathbf{y}_k$ на входной вектор $\mathbf{x}_k$. Так как векторы $\mathbf{y}_k$ и $\mathbf{x}_k$, согласно допущению, имеют одинаковые размерности, равные $m \times 1$, матрица оценки $\hat{\mathbf{M}}$ будет иметь размерность $m \times m$. Это отлично согласуется с размерностью матрицы $\mathbf{M}$, определенной выражением (2.32). Сумма в определении оценки матрицы $\hat{\mathbf{M}}$ имеет прямое отношение к матрице памяти, определенной соотношением (2.32).

Элемент внешнего произведения $\mathbf{y}_k \mathbf{x}_k^T$ обозначим $y_{ki} x_{kj}$, где x_{kj} — выходной сигнал узла j входного слоя, а y_{ki} — значение нейрона i выходного слоя. В контексте синаптического веса $w_{ij}(k)$ для k-й ассоциации узел источника j выступает в роли предсинаптического узла, а нейрон i — в качестве постсинаптического узла. Таким образом, "локальный" процесс обучения, описанный выражением (2.34), можно рассматривать как *обобщение постулата обучения Хебба* (generalization of Hebb's postulate of learning). Его также называют *правилом внешнего (матричного) произведения* (outer product rule), поскольку для построения оценки $\hat{\mathbf{M}}$ используются матричные операции. Построенная таким образом матрица памяти $\hat{\mathbf{M}}$ называется *памятью в*

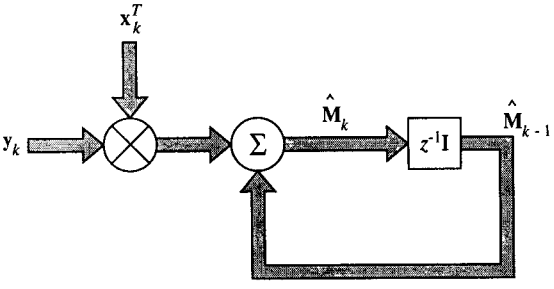


Рис. 2.19. Представление выражения (2.38) в виде графа передачи сигнала

виде матрицы корреляции (correlation matrix memory). Корреляция, в той или иной форме, является основой процесса обучения, распознавания ассоциаций и образов, а также извлечения данных из памяти в нервной системе человека [279].

Выражение (2.34) можно переписать в следующей эквивалентной форме:

$$\hat{\mathbf{M}} = [\mathbf{y}_1,\ \mathbf{y}_2,\ \ldots,\ \mathbf{y}_q] \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \ldots \\ \mathbf{x}_q^T \end{bmatrix} = \mathbf{Y}\mathbf{X}^T, \tag{2.35}$$

где

$$\mathbf{X} = [\mathbf{x}_1,\ \mathbf{x}_2,\ \ldots,\ \mathbf{x}_q]\ , \tag{2.36}$$

$$\mathbf{Y} = [\mathbf{y}_1,\ \mathbf{y}_2,\ \ldots,\ \mathbf{y}_q]\ . \tag{2.37}$$

Матрица $\mathbf{X}$ имеет размерность $m \times q$ и состоит из всего множества ключевых образов, использованных в процессе обучения. Она получила название *матрицы ключей* (key matrix). Матрица $\mathbf{Y}$ имеет размерность $m \times q$ и составлена из соответствующего множества запомненных образов. Она называется *матрицей запоминания* (memorized matrix).

Выражение (2.35) можно также записать в рекурсивной форме:

$$\hat{\mathbf{M}}_k = \hat{\mathbf{M}}_{k-1} + \mathbf{y}_k\mathbf{x}_k^T\ ,\quad k = 1,\ 2,\ \ldots,\ q. \tag{2.38}$$

Граф передачи сигнала для этого выражения показан на рис. 2.19. Согласно ему и рекурсивной формуле (2.38), матрица $\hat{\mathbf{M}}_{k-1}$ представляет собой существующую оценку матрицы памяти, а матрица $\hat{\mathbf{M}}_k$ — ее обновленную оценку в свете новой ассоциации между образами $\mathbf{x}_k$ и $\mathbf{y}_k$. Сравнивая рекурсию в формулах (2.33) и (2.38), несложно заметить, что внешнее произведение $\mathbf{y}_k\mathbf{x}_k^T$ представляет собой оценку матрицы весов $\mathbf{W}(k)$, соответствующую k-й ассоциации ключевого и запомненного образов $\mathbf{x}_k$ и $\mathbf{y}_k$.

Извлечение из памяти

Фундаментальными задачами, возникающими при использовании ассоциативной памяти, являются ее адресация и извлечение запомненных образов. Для того чтобы объяснить первый аспект этой задачи, обозначим через $\hat{\mathbf{M}}$ матрицу ассоциативной памяти, прошедшей полное обучение на q ассоциациях согласно выражению (2.34). Пусть на вход системы ассоциативной памяти подается случайный вектор возбуждения $\mathbf{x}_j$, для которого требуется получить вектор *отклика* (response):

$$\mathbf{y} = \hat{\mathbf{M}}x_j. \tag{2.39}$$

Подставляя выражение (2.34) в (2.39), получим:

$$\mathbf{y} = \sum_{k=1}^{m} \mathbf{y}_k \mathbf{x}_k^T \mathbf{x}_j = \sum_{k=1}^{m} (\mathbf{x}_k^T \mathbf{x}_j)\mathbf{y}_k, \tag{2.40}$$

где $\mathbf{x}_k^T \mathbf{x}_j$ — *скалярное произведение* $\mathbf{x}_k$ и $\mathbf{x}_j$. Выражение (2.40) можно переписать в виде

$$\mathbf{y} = (\mathbf{x}_j^T \mathbf{x}_j)\mathbf{y}_j + \sum_{k=1, k\neq j}^{m} (\mathbf{x}_k^T \mathbf{x}_j)\mathbf{y}_k. \tag{2.41}$$

Пусть ключевые образы нормализованы, т.е. имеют единичную длину (или энергию):

$$E_k = \sum_{l=1}^{m} x_{kl}^2 = \mathbf{x}_k^T \mathbf{x}_k = 1, \; k = 1, 2, \cdots q. \tag{2.42}$$

Тогда отклик памяти на возбудитель (ключевой образ) $\mathbf{x}_j$ можно упростить следующим образом:

$$\mathbf{y} = \mathbf{y}_j + \mathbf{v}_j, \tag{2.43}$$

где

$$\mathbf{v}_j = \sum_{k=1, k\neq j}^{m} (\mathbf{x}_k^T \mathbf{x}_j)\mathbf{y}_k. \tag{2.44}$$

Первое слагаемое в правой части выражения (2.43) представляет собой ожидаемый отклик $\mathbf{y}_j$. Таким образом, его можно трактовать как "сигнальную" составляющую фактического отклика $\mathbf{y}$. Второй вектор $\mathbf{v}_j$ в правой части этого выражения представляет шум, который возникает в результате смешивания ключевого вектора $\mathbf{x}_j$ со всеми остальными векторами, хранящимися в памяти. Именно вектор шума $\mathbf{v}_j$ несет ответственность за ошибки при извлечении из памяти.

В контексте линейного пространства сигналов косинус угла между векторами $\mathbf{x}_k$ и $\mathbf{x}_j$ можно определить как скалярное произведение этих векторов, деленное на произведение их Евклидовых *норм* (norm) (или длин):

$$\cos\ (\mathbf{x}_k,\ \mathbf{x}_j) = \frac{\mathbf{x}_k^T \mathbf{x}_j}{\|\mathbf{x}_k\|\ \|\mathbf{x}_j\|}. \tag{2.45}$$

$||\mathbf{x}_k||$ обозначается Евклидова норма вектора $\mathbf{x}_k$, определяемая как квадратный корень из его энергии:

$$\|\mathbf{x}_k\| = (\mathbf{x}_k^T \mathbf{x}_k)^{1/2} = E_k^{1/2}. \tag{2.46}$$

Возвращаясь к исходной задаче, заметим, что в соответствии с допущением (2.42) ключевые векторы нормализованы. Таким образом, выражение (2.45) можно упростить:

$$\cos\ (\mathbf{x}_k,\ \mathbf{x}_j) = \mathbf{x}_k^T \mathbf{x}_j. \tag{2.47}$$

Теперь можно переопределить вектор шума (2.44) следующим образом:

$$\mathbf{v}_j = \sum_{k=1, k\neq j}^{m} \cos\ (\mathbf{x}_k,\ \mathbf{x}_j)\mathbf{y}_k. \tag{2.48}$$

Известно, что если ключевые векторы являются *ортогональными* (ortogonal) (т.е. перпендикулярными друг другу в Евклидовом смысле), то

$$\cos\ (\mathbf{x}_k,\ \mathbf{x}_j) = 0, \quad k \neq j, \tag{2.49}$$

и, следовательно, вектор шума является нулевым. В этом случае отклик $\mathbf{y}$ равен $\mathbf{y}_j$. Итак, память считается *совершенно ассоциированной* (associated perfectly), если ключевые векторы выбираются из *ортогонального набора* (orthogonal set), т.е. удовлетворяют следующему условию:

$$\mathbf{x}_k^T \mathbf{x}_j = \begin{cases} 1, & k = j, \\ 0, & k \neq j. \end{cases} \tag{2.50}$$

Теперь предположим, что все ключевые векторы составляют ортогональный набор, т.е. выполняется условие (2.50). Как определить *емкость* или *запоминающую способность* (storage capacity) ассоциативной памяти? Другими словами, какое максимальное количество образов можно в ней сохранить? Ответ на этот фундаментальный вопрос связан с рангом матрицы памяти $\hat{\mathbf{M}}$. Рангом матрицы называется количество ее независимых строк (столбцов). Это значит, что если ранг прямоугольной матрицы размерности $l \times m$ равенr, то выполняется соотношение $r \leq \min(l, m)$.

В случае корреляционной памяти размерность матрицы памяти составляет $m \times m$, где m — размерность пространства входных сигналов. Отсюда следует, что ранг матрицы $\mathbf{M}$ ограничен сверху числом m. Теперь можно сформулировать утверждение о том, что количество образов, которые могут быть надежно сохранены в корреляционной памяти, не может превышать размерности пространства входных сигналов.

В реальных ситуациях часто приходится сталкиваться с тем, что ключевые образы, представленные на вход ассоциативной памяти, практически никогда не являются ортогональными. Следовательно, корреляционная память, характеризуемая соотношением (2.34), иногда может давать сбои и ошибочные результаты. Это значит, что ассоциативная память может случайно распознать и классифицировать образ, никогда ранее не виденный. Чтобы проиллюстрировать это свойство ассоциативной памяти, рассмотрим следующий набор ключевых образов:

$$\{\mathbf{x}_{\text{key}}\} : \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q,$$

и соответствующий ему набор запоминаемых образов:

$$\{\mathbf{y}_{\text{mem}}\} : \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_q.$$

Для описания близости ключевых образов в линейном пространстве сигналов введем понятие *общности* (community) и определим общность множества образов $\{\mathbf{x}_{\text{key}}\}$ как нижний предел скалярного произведения $\mathbf{x}_k^T \mathbf{x}_j$ любых двух векторов из этого множества. Пусть $\hat{\mathbf{M}}$ — матрица памяти, построенная согласно формуле (2.34) в результате обучения ассоциативной памяти на наборе ключевых образов $\{\mathbf{x}_{\text{key}}\}$ и соответствующем ему наборе откликов $\{\mathbf{y}_{\text{mem}}\}$. Отклик $\boldsymbol{y}$ этой памяти на внешнее воздействие $\mathbf{x}_j$ из набора $\{\mathbf{x}_{\text{key}}\}$ будет описываться выражением (2.39). При этом предполагается, что все векторы из множества $\{\mathbf{x}_{\text{key}}\}$ являются единичными (т.е. векторами с единичной энергией). Кроме того, предположим, что

$$\mathbf{x}_k^T \mathbf{x}_j \geq \gamma, \quad k \neq j. \tag{2.51}$$

Если нижняя граница γ достаточно велика, то память не сможет отличить вектор $\mathbf{y}$ от отклика на любой другой входной вектор из множества $\{\mathbf{x}_{key}\}$. Если ключевые образы этого набора имеют вид

$$\mathbf{x}_j = \mathbf{x}_0 + \mathbf{v}, \tag{2.52}$$

где $\mathbf{v}$ — некоторый стохастический вектор, то, вероятнее всего, память распознает вектор $\mathbf{x}_0$ и ассоциирует его с вектором $\mathbf{y}_0$, а не с вектором из фактического множества образов, использованных в процессе обучения. Здесь символами $\mathbf{x}_0$ и $\mathbf{y}_0$ обозначена никогда ранее не виденная пара сигналов. Этот феномен можно назвать *логикой животного* (которой на самом деле не существует) [210].

2.12. Адаптация

При решении реальных задач часто оказывается, что одним из основных измерений процесса обучения является пространство, а другим — время. Пространственно-временная структура обучения подтверждается многими примерами задач обучения, рассмотренными в разделе 2.10 (например, задачами управления или построения диаграммы направленности). Биологические виды, от насекомых до человека, обладают способностью для представления временной структуры опыта. Такое представление позволяет животным *адаптировать* (adapt) свое поведение к временной структуре событий в пространстве поведения [336].

Если нейронная сеть работает в *стационарной* (stationary) среде (т.е. в среде, статистические характеристики которой не изменяются во времени), она теоретически может быть обучена самым существенным статистическим характеристикам среды с помощью учителя. Например, синаптические веса сети можно вычислить в процессе обучения на множестве данных, представляющих среду. После завершения процесса обучения синаптические веса сети отражают статистическую структуру среды, которая теперь считается неизменной или "замороженной". Таким образом, для извлечения и использования накопленного опыта обучаемая система полагается на ту или иную форму *памяти*.

Однако чаще всего окружающая среда является *нестационарной* (nonstationary). Это значит, что статистические параметры входных сигналов, генерируемых средой, изменяются во времени. В такого рода ситуациях методы обучения с учителем доказали свою несостоятельность, так как сеть не обладает средствами *отслеживания* статистических вариаций среды, с которой имеет дело. Чтобы обойти этот изъян, необходимо постоянно *адаптировать* свободные параметры сети к вариациям входного сигнала в режиме *реального времени*, т.е. адаптивная система должна отвечать на каждый следующий сигнал, как на новый. Другими словами, процесс обучения в адаптивной системе не завершается, пока в нее поступают новые сигналы для обработки. Такая форма обучения называется *непрерывным обучением* (continuous learning) или *обучением на лету* (learning on-the-fly).

Для реализации непрерывного обучения можно применять *линейные адаптивные фильтры* (linear adaptive filter), построенные для линейного сумматора (т.е. отдельного нейрона, функционирующего в линейном режиме). Несмотря на довольно простую структуру (а во многом даже благодаря ей), они в настоящее время широко используются в таких несходных областях, как радиолокация, сейсмология, связь и биометрия. Теория линейных адаптивных фильтров уже достигла в своем развитии стадии зрелости [434], [1144]. Однако этого нельзя сказать о нелинейных адаптивных фильтрах[11].

[11] Задача создания оптимальиого линейного фильтра, положившая иачало теории линейных адаптивных фильтров, впервые была поставлена в [590] и независимо от нее несколько позже решеиа в [1149]. С другой стороиы, формального решения задачи оптимальиой нелинейиой фильтрации в математических терминах не

При исследовании непрерывного обучения и его применения в теории нейронных сетей возникает следующий вопрос. Как нейронная сеть может адаптировать свое поведение к изменению временной структуры входных сигналов в поведенческом пространстве? Один из ответов на этот фундаментальный вопрос предполагает, что изменения статистических характеристик нестационарных процессов протекают достаточно медленно, чтобы процесс на коротком промежутке времени можно было рассматривать как *псевдостационарный*. Приведем примеры.

- Синтез речевого сигнала можно рассматривать как стационарный процесс на интервале времени порядка 10–30 миллисекунд.
- Эхо радара от дна океана можно считать стационарным на интервалах времени порядка нескольких секунд.
- При долговременном прогнозировании погоды синоптические данные можно рассматривать как стационарные на интервалах времени порядка нескольких минут.
- В контексте оценки тенденций биржевого рынка данные можно считать стационарными на интервалах времени порядка нескольких дней.

Используя свойство псевдостационарности в стохастических процессах, можно увеличить срок эффективной работы нейронной сети за счет ее периодического *переучивания* (retraining), позволяющего учесть вариации входных данных. Такой подход можно использовать, например, для обработки биржевых данных.

Можно также применить и более точный *динамический* (dynamic) подход. Для этого нужно выполнить следующую последовательность действий.

- Выбрать достаточно короткий интервал времени, на котором данные можно считать псевдостационарными, и использовать для обучения сети.
- После получения нового обучающего примера нужно отбросить самый старый вектор и добавить в выборку новый пример.

существует. Тем не менее в 1950-х годах в этой области были опубликованы отличные работы, которые внесли некоторую ясность в природу этой задачи [1148], [1177].

Впервые идею нелинейного адаптивного фильтра выдвинул Габор в 1954 году [330]. Со своими единомышленниками он занялся созданием такого фильтра [331]. Сначала Габор предложил обойти математические трудности нелинейной адаптивной фильтрации путем создания фильтра, реакция которого оптимизируется в процессе обучения. Выходной сигнал такого фильтра выражается в виде

$$y(n) = \sum_{n=0}^{N} w_n x(n) + \sum_{n=0}^{N} \sum_{m=0}^{N} w_{n,m} x(n) x(m) + \ldots$$

где $x(0)$, $x(1)$, $\ldots$, $x(N)$ – примеры входного сигнала фильтра. (Этот полином теперь носит имя Габора–Колмогорова или называется рядом Вольтерра.) Первое слагаемое полинома представляет собой линейный фильтр, характеризуемый набором коэффициентов $\{w_n\}$. Второе слагаемое характеризуется множеством диадических коэффициентов $\{w_{n,m}\}$ и является нелинейным. Это слагаемое содержит произведение двух экземпляров входного сигнала фильтра и имеет более высокий порядок. Коэффициенты фильтра настраиваются с помощью метода градиентного спуска с целевой функцией среднеквадратического расстояния между желаемым ответом $\mathrm{d}(N)$ и фактическим выходным сигналом фильтра $\mathrm{y}(N)$.

- Использовать обновленную выборку для обучения сети.
- Непрерывно повторять описанную процедуру.

Описанный алгоритм позволяет встроить временные свойства в архитектуру нейронной сети, реализовав таким образом принцип *непрерывного обучения на упорядоченных во времени примерах* (continual learning with time-ordered examples). При использовании такого подхода нейронную сеть можно считать *нелинейным адаптивным фильтром* (nonlinear adaptive filter), представляющим собой обобщение линейного адаптивного фильтра. Однако для реализации такого динамического подхода на компьютере требуется очень высокое быстродействие, которое позволит выполнить все необходимые вычисления за один интервал дискретизации в реальном времени. Только в этом случае фильтр не будет отставать от изменения данных на входе системы.

2.13. Статистическая природа процесса обучения

В заключительном разделе настоящей главы речь пойдет о статистических аспектах обучения. В этом контексте нас не будет интересовать эволюция вектора весовых коэффициентов $\mathbf{w}$, и алгоритм обучения нейронной сети можно рассматривать как циклический. Будем оценивать только отклонение между целевой функцией $f(\mathbf{x})$ и фактической функцией $F(\mathbf{x}, \mathbf{w})$, реализованной в нейронной сети. Здесь под вектором $\mathbf{x}$ понимается входной сигнал. Это отклонение можно выразить в статистических терминах.

Нейронная сеть является всего лишь одной из форм, в которых при помощи процесса обучения можно закодировать *эмпирические знания* (empirical knowledge) о физических явлениях или окружающей среде. Под термином "эмпирические знания" подразумевается некий набор измерений, характеризующих данное явление. Чтобы конкретизировать это понятие, рассмотрим пример стохастического явления, описываемого случайным вектором $\mathbf{X}$, состоящим из набора *независимых переменных* (independent variable), и случайный скаляр D, представляющий *зависимую переменную* (dependent variable). Каждый из элементов случайного вектора $\mathbf{X}$ может иметь свой физический смысл. Предположение о том, что зависимая переменная D является скаляром, было сделано исключительно с целью упрощения изложения материала без потери общности. Предположим также, что существует N реализаций случайного вектора $\mathbf{X}$, обозначаемых $\{\mathbf{x}_i\}_{i=1}^N$, и соответствующее им множество реализаций случайного скаляра D, которое обозначим $\{d_i\}_{i=1}^N$. Эти реализации (измерения) в совокупности составляют обучающую выборку

$$\mathrm{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N. \tag{2.53}$$

Обычно мы не обладаем знаниями о функциональной взаимосвязи между $\mathbf{X}$ и D, поэтому рассмотрим модель [1133]

$$D = f(\mathbf{X}) + \varepsilon, \tag{2.54}$$

где $f(\cdot)$ — некоторая *детерминированная* (deterministic) функция векторного аргумента; ε — *ожидаемая ошибка* (expectational error), представляющая наше "незнание" зависимости между $\mathbf{X}$ и D. Статистическая модель, описанная выражением (2.54), называется *регрессионной* (regression model) (рис. 2.20, *а*). Ожидаемая ошибка ε в общем случае является случайной величиной с нормальным распределением и нулевым математическим ожиданием. Исходя из этого, регрессионная модель на рис. 2.20, *а* обладает двумя важными свойствами.

1. *Среднее значение ожидаемой ошибки* ε *для любой реализации* $\mathbf{x}$ *равно нулю*, т.е.

$$E[\varepsilon|\mathbf{x}] = 0, \tag{2.55}$$

где E — статистический оператор математического ожидания. Естественным следствием этого свойства является утверждение о том, что *регрессионная функция* $f(\mathbf{x})$ *является условным средним модели выхода* D *для входного сигнала* $\mathbf{X} = \mathbf{x}$:

$$f(\mathbf{x}) = E[D|\mathbf{x}]. \tag{2.56}$$

Это свойство непосредственно следует из выражения (2.54) в свете (2.55).

2. *Ожидаемая ошибка* ε *не коррелирует с функцией регрессии* $f(\mathbf{X})$, т.е.

$$E[\varepsilon f(\mathbf{X})] = 0. \tag{2.57}$$

Это свойство широко известно как *принцип ортогональности* (principle of orthogonality), который гласит, что вся информация о D, доступная через входной сигнал $\mathbf{X}$, закодирована в функции регрессии $f(\mathbf{X})$. Равенство (2.57) без труда иллюстрируется следующими выкладками:

$$E[\varepsilon f(\mathbf{X})] = E[E[\varepsilon f(\mathbf{X})|\mathbf{x}]] = E\quad[f(\mathbf{X})E[\varepsilon|\mathbf{x}]] = E[f(\mathbf{X}) \cdot 0] = 0.$$

Регрессионная модель (рис. 2.20, *а*) представляет собой математическое описание стохастической среды. В ней вектор $\mathbf{X}$ используется для описания или предсказания зависимой переменной D. На рис. 2.20, *б* представлена соответствующая "физическая" модель данной среды. Эта вторая модель, основанная на нейронной сети, позволяет закодировать эмпирические знания, заключенные в обучающей выборке $\mathbf{T}$, с помощью соответствующего набора векторов синаптических весов $\mathbf{w}$:

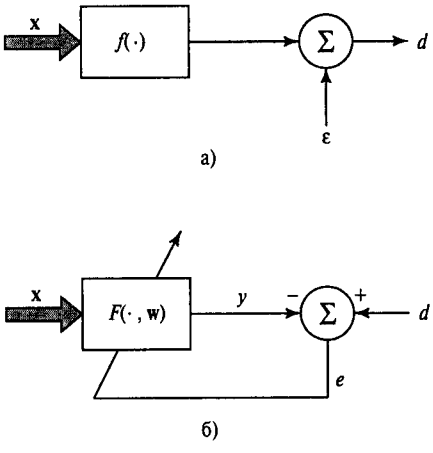


Рис. 2.20. Математическое (а) и физическое (б) представление нейронной сети

$$\mathrm{T} \rightarrow \mathbf{w}. \tag{2.58}$$

Таким образом, нейронная сеть обеспечивает аппроксимацию регрессионной модели, представленной на рис. 2.20, *а*. Пусть фактический отклик нейронной сети на входной вектор **x** обозначается следующей вероятностной переменной:

$$Y = F(\mathbf{X}, \mathbf{w}). \tag{2.59}$$

где $F(\cdot, \mathbf{w})$ — функция отображения входных данных в выходные, реализуемая с помощью нейронной сети. Для набора данных обучения T, представленного в виде множества (2.53), вектор синаптических весов **w** можно вычислить путем минимизации функции стоимости:

$$\mathbf{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(d_i - F(\mathbf{x}_i, \mathbf{w}))^2, \tag{2.60}$$

где коэффициент 1/2 вводится исключительно из соображений совместимости с обозначениями, которые использовались ранее и будут использоваться в следующих главах. Если не принимать во внимание коэффициент 1/2, функция стоимости E(**w**) описывает сумму квадратов разностей между желаемым d и фактическим y откликами нейронной сети для всего набора примеров обучения T. Использование соотношения (2.60) в качестве функции стоимости отражает "пакетный" характер обучения. Это значит, что настройка синаптических весов нейронной сети выполняется для всего массива примеров обучения в целом, а не для каждого примера в отдельности.

Пусть E_{T} — *оператор усреднения* (average operator) по всей обучающей выборке T. Переменные, или их функции, обрабатываемые оператором усреднения E_{T}, обозначим символами $\mathbf{x}$ и d. При этом пара $(\mathbf{x}, d)$ представляет каждый конкретный обучающий пример из набора T. В отличие от оператора усреднения оператор статистического ожидания E функционирует на множестве всех значений случайных переменных $\mathbf{X}$ и D, подмножеством которого является T. Различие между операторами E_{T} и E будет четко показано ниже.

В свете преобразования, описываемого формулой (2.58), функции $F(\mathbf{x}, \mathbf{w})$ и $F(\mathbf{x}, \mathrm{T})$ являются взаимозаменяемыми, поэтому выражение (2.60) можно переписать в виде

$$\mathbf{E}(\mathbf{w}) = \frac{1}{2}\,\mathrm{E}_{\mathrm{T}}[(d_i - F(\mathbf{x}_i,\ \mathrm{T}))^2]. \tag{2.61}$$

Добавляя функцию $f(\mathbf{x})$ к аргументу $(d_i - F(\mathbf{x}_i,\ \mathrm{T}))^2$ и вычитая ее, а затем используя (2.54), получим:

$$d - F(\mathbf{x}, \mathrm{T}) = (d - f(\mathbf{x})) + (f(\mathbf{x}) - F(\mathbf{x}, \mathrm{T})) = \varepsilon + (f(\mathbf{x}) - F(\mathbf{x}, \mathrm{T})).$$

Подставляя это выражение в (2.61) и раскрывая скобки, функцию стоимости можно записать в следующей эквивалентной форме:

$$\mathbf{E}(\mathbf{w}) = \frac{1}{2}E_{\mathrm{T}}[\varepsilon^2] + \frac{1}{2}E_{\mathrm{T}}[(f(\mathbf{x}) - F(\mathbf{x},\ \mathrm{T}))^2] + E_{\mathrm{T}}[\varepsilon(f(\mathbf{x}) - F(\mathbf{x},\ \mathrm{T}))]. \tag{2.62}$$

Заметим, что последнее слагаемое в правой части формулы (2.62) равно нулю по двум причинам.

Ожидаемая ошибка ε не коррелирует с регрессионной функцией $f(\mathbf{x})$, что видно из выражения (2.57), интерпретируемого в терминах оператора E_{T}.

Ожидаемая ошибка ε относится к регрессионной модели, изображенной на рис. 2.20, *а*, в то время как аппроксимирующая функция $F(\mathbf{x}, \mathbf{w})$ относится к нейросетевой модели, показанной на рис. 2.20, *б*.

Следовательно, выражение (2.62) можно упростить:

$$\mathbf{E}(\mathbf{w}) = \frac{1}{2}E_{\mathrm{T}}[\varepsilon^2] + \frac{1}{2}E_{\mathrm{T}}[(f(\mathbf{x}) - F(\mathbf{x},\ \mathrm{T}))^2]. \tag{2.63}$$

Первое слагаемое в правой части выражения (2.63) описывает дисперсию ожидаемой ошибки (регрессионного моделирования) ε, вычисленной на обучающей выборке T. Это *исходная* (intrinsic) ошибка, так как она не зависит от вектора весов $\mathbf{w}$. Ее можно не учитывать, так как главной задачей является минимизация функции стоимости $\mathbf{E}(\mathbf{w})$ относительно вектора $\mathbf{w}$. Следует учитывать, что значение вектора весов $\mathbf{w}^*$,

минимизирующее функцию стоимости $\mathbf{E}(\mathbf{w})$, будет также минимизировать и среднее по ансамблю квадратичное расстояние между регрессионной функцией $f(\mathbf{x})$ и функцией аппроксимации $F(\mathbf{x}, \mathbf{w})$. Другими словами, естественной мерой эффективности использования $F(\mathbf{x}, \mathbf{w})$ для прогнозирования желаемого отклика d является следующая функция:

$$L_{\text{av}}(f(\mathbf{x}),\ F(\mathbf{x}, \mathbf{w})) = \mathrm{E}_{\mathrm{T}}[(f(\mathbf{x}) - F(\mathbf{x},\ \mathrm{T}))^2]. \tag{2.64}$$

Этот результат имеет фундаментальное значение, так как он обеспечивает математическую основу для изучения зависимости между смещением и дисперсией, полученными в результате использования $F(\mathbf{x}, \mathbf{w})$ в качестве аппроксимации функции $f(\mathbf{x})$ [344].

Дилемма смещения и дисперсии

Используя формулу (2.56), квадрат расстояния между функциями $F(\mathbf{x}, \mathbf{w})$ и $f(\mathbf{x})$ можно переопределить в виде

$$L_{\text{av}}(f(\mathbf{x}),\ F(\mathbf{x}, \mathbf{w})) = \mathrm{E}_{\mathrm{T}}[(E[D|\mathbf{X} = \mathbf{x}] - F(\mathbf{x},\ \mathrm{T}))^2]. \tag{2.65}$$

Это выражение также можно рассматривать как среднее значение *ошибки оценивания* (estimation error) регрессионной функции $f(\mathbf{x})$=$E[D|\mathbf{X}$=$\mathbf{x}]$ функцией аппроксимации $F(\mathbf{x}, \mathbf{w})$, вычисленной на всем обучающем множестве T. Обратите внимание, что условное среднее $E[D|\mathbf{X}$=$\mathbf{x}]$ имеет постоянное математическое ожидание на обучающем множестве T. Далее, добавляя и вычитая среднее $E_{\mathrm{T}}[F(\mathbf{x}, \mathrm{T})]$, получим:

$$E[D|\mathbf{X} = \mathbf{x}] - F(\mathbf{x}, \mathrm{T}) = (E[D|\mathbf{X} = \mathbf{x}] - E_{\mathrm{T}}[F(\mathbf{x}, \mathrm{T})]) + (E_{\mathrm{T}}[F(\mathbf{x}, \mathrm{T})] - F(\mathbf{x}, \mathrm{T})).$$

Выполняя преобразования, использованные для вывода соотношения (2.62) из выражения (2.61), формулу (2.65) можно представить в виде суммы двух слагаемых (см. задачу 2.22):

$$L_{\text{av}}(f(\mathbf{x}),\ F(\mathbf{x}, \mathbf{T})) = B^2(\mathbf{w}) + V(\mathbf{w}), \tag{2.66}$$

где $B(\mathbf{w})$ и $V(\mathbf{w})$ определяются следующим образом:

$$B(\mathbf{w}) = \mathrm{E}_{\mathrm{T}}[F(\mathbf{x},\ \mathrm{T})] - \mathrm{E}[D|\ \mathbf{X} = \mathbf{x}], \tag{2.67}$$

$$V(\mathbf{w}) = \mathrm{E}_{\mathrm{T}}[(F(\mathbf{x},\ \mathrm{T}) - \mathrm{E}_{\mathrm{T}}[F(\mathbf{x},\ \mathrm{T})])^2]. \tag{2.68}$$

Теперь можно сформулировать два важных наблюдения.

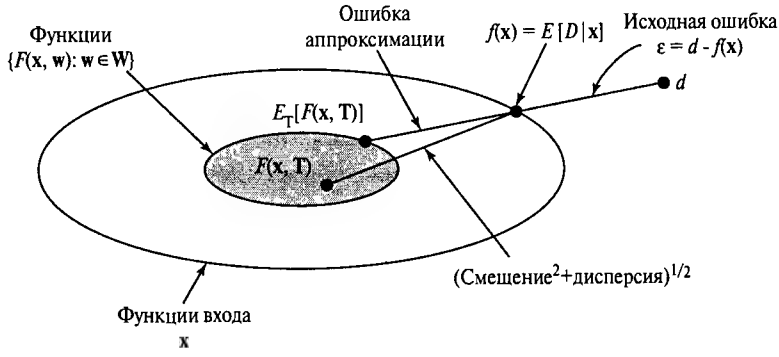


Рис. 2.21. Различные источники ошибки при решении задачи регрессии

1. Элемент $B(\mathbf{w})$ описывает *смещение* (bias) среднего значения аппроксимирующей функции $F(\mathbf{x}, \mathrm{T})$ относительно регрессионной функции $f(\mathbf{x})=E[D|\mathbf{X}=\mathbf{x}]$. Этот элемент выражает неспособность нейронной сети, представленной функцией $F(\mathbf{x}, \mathbf{w})$, точно аппроксимировать регрессионную функцию $f(\mathbf{x})=E[D|\mathbf{X}=\mathbf{x}]$. Таким образом, элемент $B(\mathbf{w})$ можно считать *ошибкой аппроксимации* (approximation error).
2. Элемент $V(\mathbf{w})$ представляет *дисперсию* (variance) аппроксимирующей функции $F(\mathbf{x}, \mathbf{w})$ на всем обучающем множестве T. Это слагаемое отражает неадекватность информации о регрессионной функции $f(\mathbf{x})$, содержащейся в обучающем множестве T. Таким образом, элемент $V(\mathbf{w})$ можно считать *ошибкой оценивания* (estimation error).

На рис. 2.21 изображены взаимосвязи между целевой и аппроксимирующей функциями, а также наглядно показано, как накапливаются ошибки оценивания — смещение и дисперсия. Чтобы достичь хорошей производительности, смещение $B(\mathbf{w})$ и дисперсия $V(\mathbf{w})$ функции аппроксимации $F(\mathbf{x}, \mathbf{w})=F(\mathbf{x}, \mathrm{T})$ должны быть невелики.

К сожалению, в нейронных сетях, обучаемых на данных выборки фиксированного размера, малое смещение достигается за счет большой дисперсии. Одновременно уменьшить дисперсию и смещение можно только в одном случае — если размер обучающего множества бесконечно велик. Эта проблема называется *дилеммой смещения/дисперсии* (bias/variance dilemma). Следствием этой проблемы является медленная сходимость процесса обучения [344]. Дилемму смещения/дисперсии можно обойти, если преднамеренно ввести такое смещение, которое сводит дисперсию на нет или значительно ее уменьшает. Однако при этом необходимо убедиться в том, что встроенное в нейронную сеть смещение является приемлемым. Например, в контексте классификации образов смещение можно считать "приемлемым", если оно оказывает сильное влияние на среднеквадратическую ошибку только в случае попытки вывода регрессии, которая не принадлежит ожидаемому классу. В общем случае смещение необходимо *задавать* отдельно для каждой предметной области. На практике для

достижения этой цели используют *ограниченную* (constrained) сетевую архитектуру, которая обычно работает лучше, чем архитектура общего назначения. В частности, ограничения (а значит и смещение) могут принимать форму априорных знаний, встроенных в архитектуру нейронной сети путем *совместного использования весов* (если несколько синапсов сети находится под управлением одного и того же весового коэффициента связи) и/или создания *локальных рецепторных полей* (local receptive field), связанных с отдельными нейронам сети (что было продемонстрировано при использовании многослойного персептрона для решения задачи распознавания оптических символов) [621]. Подобные архитектуры уже кратко описывались в разделе 1.7.

2.14. Теория статистического обучения

В этом разделе мы продолжим рассмотрение статистических свойств нейронных сетей. Основное внимание будет уделено *теории обучения* (learning theory), связанной с решением фундаментального вопроса о том, как управлять обобщающей способностью нейронных сетей. Этот вопрос будет рассматриваться в контексте обучения с учителем.

Модель обучения с учителем состоит из трех взаимосвязанных компонентов (рис. 2.22). В математических терминах они описываются следующим образом [1084], [1086].

1. *Среда* (environment). Среда является стационарной. Она представлена векторами $\mathbf{x}$ с фиксированной, но неизвестной функцией распределения вероятности $F_X(\mathbf{x})$.
2. *Учитель* (teacher). Учитель генерирует желаемый отклик d для каждого из входных векторов $\mathbf{x}$, полученных из внешней среды, в соответствии с условной функцией распределения $F_X(\mathbf{x}|d)$, которая тоже фиксирована, но неизвестна. Желаемый отклик d и входной вектор $\boldsymbol{x}$ связаны следующим соотношением:

$$d = f(\mathbf{x}, v), \tag{2.69}$$

   где v — шум, т.е. изначально предполагается "зашумленность" данных учителя.
3. *Обучаемая машина* (learning machine). Обучаемая машина (нейронная сеть) "способна реализовать множество функций отображения" вход-выход, описываемых соотношением

$$y = F(\mathbf{x}, \mathbf{w}), \tag{2.70}$$

   где y — фактический отклик, сгенерированный обучаемой машиной в ответ на входной сигнал $\mathbf{x}$; $\mathbf{w}$ — набор свободных параметров (синаптических весов), выбранных из пространства параметров $\mathbf{W}$.

Уравнения (2.69) и (2.70) записаны в терминах примеров, используемых при обучении.

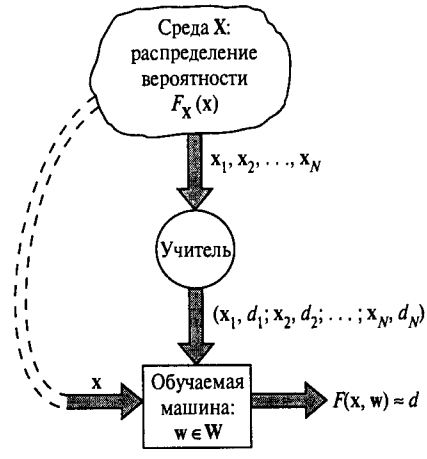


Рис. 2.22. Модель процесса обучения с учителем

Задача обучения с учителем состоит в выборе конкретной функции $F(\mathbf{x}, \mathbf{w})$, которая оптимально (в некотором статистическом смысле) аппроксимирует ожидаемый отклик d. Выбор, в свою очередь, основывается на множестве N *независимых, равномерно распределенных* примеров обучения, описываемых формулой (2.53). Для удобства изложения материала воспроизведем эту запись еще раз:

$$\mathrm{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N}.$$

Каждая пара выбирается обучаемой машиной из множества **T** с некоторой обобщенной функцией распределения вероятности $F_{X,D}(\mathbf{x}, d)$, которая, как и другие функции распределения, фиксирована, но неизвестна. Принципиальная возможность обучения с учителем зависит от ответа на следующий вопрос: содержат ли примеры из множества $\{(\mathbf{x}_i, d_i)\}$ достаточно информации для создания обучаемой машины, обладающей хорошей обобщающей способностью? Ответ на этот вопрос лежит в области результатов, впервые полученных в 1971 году [1088]. Поэтому рассмотрим задачу обучения с учителем как *задачу аппроксимации* (approximation problem), состоящую в нахождении функции $F(\mathbf{x}, \mathbf{w})$, которая наилучшим образом приближает желаемую функцию $f(\mathbf{x})$.

Пусть $L(d, F(\mathbf{x}, \mathbf{w}))$ — мера *потерь* или *несходства* между желаемым откликом d, соответствующим входному вектору $\mathbf{x}$, и откликом $F(\mathbf{x}, \mathbf{w})$, сгенерированным обучаемой машиной. В качестве меры $L(d, F(\mathbf{x}, \mathbf{w}))$ часто рассматривают *квадратичную функцию потерь* (quadratic loss function), определенную как квадрат расстояния между $d = f(\mathbf{x})$ и аппроксимацией $F(\mathbf{x}, \mathbf{w})$[12],

[12] Функция стоимости $L(d, F(\mathbf{x}, \mathbf{w}))$, определяемая формулой (2.71), применяется к скаляру d. В случае использования вектора $\mathbf{d}$ в качестве желаемого отклика функция аппроксимации $F(\mathbf{x}, \mathbf{w})$ является векторной. Тогда в качестве функции потерь выбирается квадрат Евклидова расстояния $L(\mathbf{d}, \mathbf{F}(\mathbf{x}, \mathbf{w})) = ||\mathbf{d} - \mathbf{F}(\mathbf{x}, \mathbf{w})||^2$, где $\mathbf{F}(\cdot, \cdot)$ — вектор-функция своих аргументов.

$$L(d, F(\mathbf{x}, \mathbf{w})) = (d - F(\mathbf{x}, \mathbf{w}))^2. \tag{2.71}$$

Квадратичное расстояние в формуле (2.64) — это усредненное по множеству всех пар примеров ($\mathbf{x}$, d) расширение меры $L(d, F(\mathbf{x}, \mathbf{w}))$.

В литературе, посвященной теории статистического обучения, обычно рассматриваются конкретные функции потерь. Основное свойство представленной здесь теории статистического обучения состоит в том, что форма функции потерь $L(d, F(\mathbf{x}, \mathbf{w}))$ не играет особой роли. Конкретный вид функции потерь будет рассмотрен несколько позже в этом разделе.

Ожидаемая величина потерь определяется *функционалом риска* (risk functional)

$$R(\mathbf{w}) = \int L(d,\ F(\mathbf{x},\ \mathbf{w}))\ dF_{\mathbf{X},\ D}(\mathbf{x},\ d), \tag{2.72}$$

где интеграл берется по всем возможным значениям ($\mathbf{x}$, d). Целью обучения с учителем является минимизация функционала риска $R(\mathbf{w})$ в классе функций аппроксимации $\{F(\mathbf{x}, \mathbf{w}), \mathbf{w} \in \mathbf{W}\}$. Однако оценка функционала риска усложняется тем, что обобщенная функция распределения $F_{X,D}(\mathbf{x}, d)$ обычно неизвестна. При обучении с учителем вся доступная информация содержится в множестве данных обучения T. Чтобы обойти эту математическую сложность, будем использовать индуктивный принцип минимизации эмпирического риска [1087]. Этот принцип основан на доступности обучающего множества T, что идеально согласуется с философией нейронных сетей.

Некоторые основные определения

Прежде чем двигаться дальше, введем несколько основных определений, которые будут использоваться в изложении последующего материала.

Сходимость по вероятности. Рассмотрим последовательность случайных переменных $a_1, a_2, \ldots, a_N$. Эта последовательность считается *сходящейся по вероятности* (converge in probability) к случайной переменной a_0, если для любого $\sigma > 0$ выполняется следующее вероятностное соотношение:

$$P(|a_N - a_0| > \sigma) \xrightarrow{P} \text{при} \quad N \to \infty. \tag{2.73}$$

Нижний и верхний пределы (supremum & infimum). Верхним пределом непустого множества скалярных величин **A** (sup **A**) называется наименьший из скаляров x, для которых истинно неравенство $x \geq y$ для всех $y \in \mathbf{A}$. Если такой скалярной величины не существует, то считается, что верхним пределом непустого множества **A** является бесконечность. Аналогично, нижним пределом непустого множества скаляров **A** (inf **A**) называется наибольший из скаляров x, для которых истинно неравенство $x \leq y$ для всех $y \in \mathbf{A}$. Если такой скалярной величины не существует, то считается, что нижним пределом непустого множества **A** является бесконечность.

Функционал эмпирического риска. Для обучающего множества $\mathbf{T} = \{(\mathbf{x}_i,\ d_i)\}_{i=1}^{N}$ функционал эмпирического риска определяется в терминах функции потерь $L(d_i, F(\mathbf{x}_i, \mathbf{w}_i))$ следующим образом:

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(d_i,\ F(\mathbf{x}_i,\ \mathbf{w})). \tag{2.74}$$

Строгая состоятельность (strict consistency). Рассмотрим множество $\mathbf{W}$ функций $L(d, F(\mathbf{x}, \mathbf{w}))$, распределение которых определяется интегральной функцией распределения $F_{X,D}(\mathbf{x}, d)$. Пусть $\mathbf{W}(c)$ — непустое подмножество этого множества, такое, что

$$\mathbf{W}(c) = \left\{\mathbf{w} : \int L(d, F(\mathbf{x}, \mathbf{w})) \geq c\right\}, \tag{2.75}$$

где $c \in (-\infty,\ +\infty)$. Функционал эмпирического риска считается *строго состоятельным* (strictly consistent), если для любого подмножества $\mathbf{W}(c)$ обеспечивается сходимость по вероятности

$$\inf_{\mathbf{w}\in\mathbf{W}(c)} R_{\text{emp}}(\mathbf{w}) \xrightarrow{P} \inf_{\mathbf{w}\in\mathbf{W}(c)} R(\mathbf{w}) \quad \text{при} \quad N \to \infty. \tag{2.76}$$

Теперь, ознакомившись с этими определениями, можно продолжить изучение теории статистического обучения Вапника (Vapnik).

Принцип минимизации эмпирического риска

Основная идея принципа *минимизации эмпирического риска* (empirical risk minimization) состоит в использовании функционала эмпирического риска $R_{\text{emp}}(\mathbf{w})$, определяемого формулой (2.74). Этот новый функционал отличается от функционала $R(\mathbf{w})$, задаваемого формулой (2.72), в двух аспектах.

1. Он явно *не* зависит от неизвестной функции распределения $F_{X,D}(\mathbf{x}, d)$.
2. Теоретически его можно минимизировать по вектору весовых коэффициентов $\mathbf{w}$.

Пусть $\mathbf{w}_{\text{emp}}$ и $F(\mathbf{x}, \mathbf{w}_{\text{emp}})$ — вектор весов и соответствующее ему отображение, которые минимизируют функционал эмпирического риска $R_{\text{emp}}(\mathbf{w})$, определяемый формулой (2.74). Аналогично, пусть $\mathbf{w}_o$ и $F(\mathbf{x}, \mathbf{w}_o)$ — вектор весовых коэффициентов и отображение, минимизирующие фактический функционал риска $R(\mathbf{w})$, заданный формулой (2.72). Векторы $\mathbf{w}_{\text{emp}}$ и $\mathbf{w}_o$ принадлежат пространству весов $\mathbf{W}$. Требуется найти условия, при которых аппроксимирующее отображение $F(\mathbf{x}, \mathbf{w}_{\text{emp}})$ достаточно "близко" к фактическому отображению $F(\mathbf{x}, \mathbf{w}_o)$ (в качестве меры близости будем использовать разницу между $R_{\text{emp}}(\mathbf{w})$ и $R(\mathbf{w})$).

Для некоторого фиксированного $\mathbf{w} = \mathbf{w}^*$ функционал риска $R(\mathbf{w}^*)$ определяет *математическое ожидание* случайной переменной, определяемое соотношением

$$Z_{\mathbf{w}^*} = L(d,\ F(\mathbf{x},\ \mathbf{w}^*)). \tag{2.77}$$

В отличие от него функционал эмпирического риска $R_{\text{emp}}(\mathbf{w}^*)$ обеспечивает *эмпирическое (арифметическое) среднее значение* (empirical (arithmetic) mean) случайной переменной Z_{w^*}. Согласно *закону больших чисел* (law of large numbers), который составляет одну из основных теорем теории вероятностей, для обучающего множества T бесконечно большого размера N эмпирическое среднее случайной переменной Z_{w^*} в общем случае сходится к ее ожидаемому значению. Это наблюдение обеспечивает теоретический базис для использования функционала эмпирического риска $R_{\text{emp}}(\mathbf{w})$ вместо функционала риска $R(\mathbf{w})$. Однако тот факт, что эмпирическое среднее переменной Z_{w^*} сходится к ее ожидаемому значению, совершенно не означает, что вектор весовых коэффициентов $\mathbf{w}_{\text{emp}}$, минимизирующий функционал эмпирического риска $R_{\text{emp}}(\mathbf{w})$, будет также минимизировать и функционал риска $R(\mathbf{w})$.

Этому требованию можно приближенно удовлетворить, применив следующий подход. Если функционал эмпирического риска $R_{\text{emp}}(\mathbf{w})$ *аппроксимирует* исходный функционал риска $R(\mathbf{w})$ *равномерно* по $\mathbf{w}$ с некоторой *точностью* ε, то минимум $R_{\text{emp}}(\mathbf{w})$ отстоит от минимума $R(\mathbf{w})$ не более чем на величину 2ε. Формально это означает необходимость обязательного выполнения следующего условия. Для любого $\mathbf{w} \in \mathbf{W}$ и $\varepsilon > 0$ должно выполняться вероятностное соотношение [1087]

$$P(\sup_{\mathbf{w}} |\ R(\mathbf{w})\ -\ R_{\text{emp}}(\mathbf{w})| > \varepsilon) \to 0 \text{ при } N \to \infty. \tag{2.78}$$

Если выполняется условие (2.78), то можно утверждать, что *вектор весов* $\mathbf{w}$ *среднего эмпирического риска равномерно сходится к своему ожидаемому значению*. Таким образом, если для любой наперед заданной точности ε и некоторого положительного α выполняется неравенство

$$P(\sup_{\mathbf{w}} |\ R(\mathbf{w})\ -\ R_{emp}(\mathbf{w})| > \varepsilon) < \alpha, \tag{2.79}$$

то выполняется также и следующее неравенство:

$$P(R(\mathbf{w}_{\text{emp}}) - R(\mathbf{w}_o) > 2\varepsilon) < \alpha. \tag{2.80}$$

Другими словами, если выполняется условие (2.79), то с вероятностью $(1 - \alpha)$ решение $F(\mathbf{x}, \mathbf{w}_{emp})$, минимизирующее функционал эмпирического риска $R_{\text{emp}}(\mathbf{w})$, обеспечивает отличие фактического риска $R(\mathbf{w}_{\text{emp}})$ от минимально возможного фактического риска на величину, не превышающую 2ε. Это значит, что при выполнении (2.79) с вероятностью $(1 - \alpha)$ одновременно выполняются следующие два неравенства:

$$R(\mathbf{w}_{\text{emp}}) - R_{\text{emp}}(\mathbf{w}_{\text{emp}}) < \varepsilon, \tag{2.81}$$

$$R_{\text{emp}}(\mathbf{w}_o) - R(\mathbf{w}_o) < \varepsilon. \tag{2.82}$$

Эти два соотношения определяют различие между функционалами истинного и эмпирического рисков в точках $\mathbf{w}$=$\mathbf{w}_{\text{emp}}$ и $\mathbf{w}$=$\mathbf{w}_o$. Учитывая, что $\mathbf{w}_{\text{emp}}$ и $\mathbf{w}_o$ являются точками минимума функционалов $R_{\text{emp}}(\mathbf{w})$ и $R(\mathbf{w})$, можно сделать вывод о том, что

$$R_{\text{emp}}(\mathbf{w}_{\text{emp}}) \leq R_{\text{emp}}(\mathbf{w}_o). \tag{2.83}$$

Складывая неравенства (2.81) и (2.82) и принимая во внимание неравенство (2.83), можно записать:

$$R(\mathbf{w}_{\text{emp}}) - R(\mathbf{w}_o) < 2\varepsilon. \tag{2.84}$$

Поскольку неравенства (2.81) и (2.82) одновременно выполняются с вероятностью $(1 - \alpha)$, то с такой же вероятностью выполняется и неравенство (2.84). Также можно утверждать, что с вероятностью α будет выполняться неравенство

$$R(\mathbf{w}_{\text{emp}}) - R(\mathbf{w}_o) > 2\varepsilon,$$

которое является еще одной формой утверждения (2.80).

Теперь можно формально сформулировать *принцип минимизации эмпирического риска* (principle of empirical risk minimization), состоящий из трех взаимосвязанных частей [1084], [1087].

1. Вместо функционала риска $R(\mathbf{w})$ строится функционал эмпирического риска

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(d_i, F(\mathbf{x}_i, \mathbf{w}))$$

на базе множества примеров обучения $(\mathbf{x}_i, d_i), i = 1, 2, \ldots, N$.

2. Пусть $\mathbf{w}_{\text{emp}}$ — вектор весовых коэффициентов, минимизирующий функционал эмпирического риска $R_{emp}(\mathbf{w})$ в пространстве весов $\mathbf{W}$. Тогда $R(\mathbf{w}_{\text{emp}})$ сходится по вероятности к минимально возможным значениям фактического риска $R(\mathbf{w})$, $\mathbf{w}\in\mathbf{W}$. При этом при увеличении количества N примеров обучения до бесконечности функционал эмпирического риска $R_{\text{emp}}(\mathbf{w})$ равномерно сходится к функционалу фактического риска $R(\mathbf{w})$.

3. Равномерная сходимость, определяемая как

$$P(\sup_{\mathbf{w}\in\mathbf{W}} | R(\mathbf{w}) - R_{\text{emp}}(\mathbf{w})| > \varepsilon) \to 0 \text{ при } N \to \infty,$$

является необходимым и достаточным условием непротиворечивости принципа минимизации эмпирического риска.

Для физической интерпретации этого важного принципа проведем следующее наблюдение. До обучения машины все аппроксимирующие функции равноценны. По мере обучения правдоподобие тех функций аппроксимации, которые не противоречат обучающему множеству $\{(\mathbf{x}_i,\ d_i)\}_{i=1}^{N}$, возрастает. По мере увеличения количества использованных при обучении примеров и, следовательно, повышения "плотности" входного пространства точка минимума функционала эмпирического риска $R_{\text{emp}}(\mathbf{w})$ сходится по вероятности к точке минимума функционала фактического риска $R(\mathbf{w})$.

VC-измерение

Теория равномерной сходимости функционала эмпирического риска $R_{\text{emp}}(\mathbf{w})$ к функционалу фактического риска $R(\mathbf{w})$ включает ограничения на скорость сходимости, которые основаны на важном параметре, получившем название *измерения Вапника–Червоненкиса* (Vapnik-Chervonenkis dimension) (или просто *VC-измерения*) в честь ученых, которые в 1971 году ввели это понятие [1088]. Измерение Вапника–Червоненкиса является мерой *емкости* (capacity) или *вычислительной мощности* семейства функций классификации, реализованных обучаемыми машинами.

Для того чтобы описать концепцию VC-измерения в ракурсе изучаемой проблемы, рассмотрим задачу двоичной классификации образов, для которой множество ожидаемых откликов состоит всего из двух значений $d \in \{0, 1\}$. Для обозначения правила принятия решения или функции двоичной классификации будем использовать термин *дихотомия* (dichotomy). Пусть $\mathbf{F}$ — множество дихотомий, реализованных обучаемой машиной, т.е.

$$\mathbf{F} = \{F(\mathbf{x}, \mathbf{w}) : \mathbf{w} \in \mathbf{W}, F : \Re^m \mathbf{W} \to \{0, 1\}\}. \tag{2.85}$$

Пусть $\mathbf{L}$ — множество, содержащее N точек m-мерного пространства $\mathbf{X}$ входных векторов:

$$\mathbf{L} = \{\mathbf{x}_i \in \mathbf{X}; i = 1, 2, \ldots, N\}. \tag{2.86}$$

Дихотомия, реализованная обучаемой машиной, разбивает множество $\mathbf{L}$ на два непересекающихся подмножества $\mathbf{L}_0$ и $\mathbf{L}_1$, таких, что

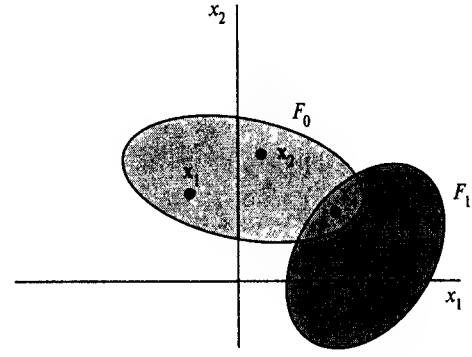


Рис. 2.23. Диаграмма для примера 2.1

$$F(\mathbf{x}, \mathbf{w}) = \begin{cases} 0 & \text{для } \mathbf{x} \in \mathbf{L}_0, \\ 1 & \text{для } \mathbf{x} \in \mathbf{L}_1. \end{cases} \tag{2.87}$$

Пусть $\Delta_F(\mathbf{L})$ — количество различных дихотомий, реализованных обучаемой машиной; $\Delta_F(l)$ — максимум $\Delta_F(\mathbf{L})$ на множестве всех $\mathbf{L}$, для которых $|\mathbf{L}| = l$, где $|\mathbf{L}|$ — количество элементов в $\mathbf{L}$. Говорят, что ансамбль дихотомий $\mathbf{F}$ является *разбиением* множества $\mathbf{L}$, если $\Delta_F(\mathbf{L})=2^{|L|}$, т.е. если все возможные дихотомии в $\mathbf{L}$ могут быть реализованы функциями $\mathbf{F}$. При этом $\Delta_F(l)$ называется *функцией роста* (growth function).

Пример 2.1

На рис. 2.23 показано двумерное входное пространство $\mathbf{X}$, состоящее из четырех точек $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$. Показанные на рисунке границы решений функций F_0 и F_1 соответствуют классам (гипотезам) 0 и 1. На рис. 2.23 видно, что функция F_0 индуцирует дихотомию

$$\mathbf{D}_0 = \{\mathbf{G}_0 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4\}, \mathbf{G}_1 = \{\mathbf{x}_3\}\}.$$

С другой стороны, функция F_1 описывает дихотомию

$$\mathbf{D}_1 = \{\mathbf{G}_0 = \{\mathbf{x}_1, \mathbf{x}_2\}, \mathbf{G}_1 = \{\mathbf{x}_3, \mathbf{x}_4\}\}.$$

Так как множество $\mathbf{G}$ состоит из четырех точек, мощность $|\mathbf{G}| = 4$. Следовательно,

$$\Delta_F(\mathbf{G}) = 2^4 = 16.$$

■

Возвращаясь к общему обсуждению ансамбля дихотомий $\mathbf{F}$, описываемого формулой (2.85), и соответствующего множества точек $\boldsymbol{L}$, задаваемого формулой (2.86), VC-измерение можно определить следующим образом [551], [1084], [1088], [1094].

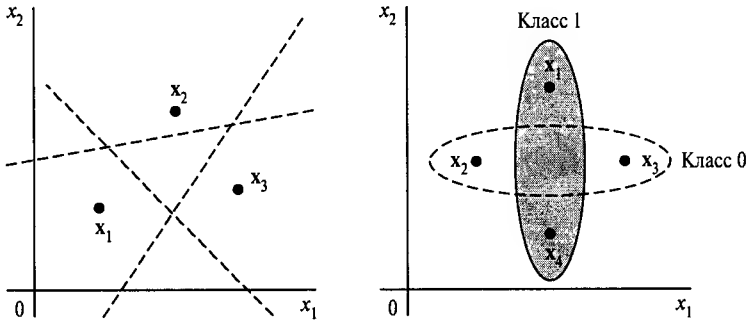


Рис. 2.24. Два двумерных распределения для примера 2.2

VC-измерением $\mathbf{F}$ *называется мощность наибольшего множества* $\mathbf{L}$*, разбиением которого является* $\mathbf{F}$*.*

Другими словами, VC-измерением $\mathbf{F}$ (обозначается VCdim($\mathbf{F}$)) является самое большое значение N, для которого $\Delta_F(N) = 2^N$. В более знакомых терминах это утверждение можно переформулировать следующим образом. VC-измерение множества функций классификации $\{F(\mathbf{x}, \mathbf{w})$: $\mathbf{w} \in \mathbf{W}\}$ — это максимальное число образов, на которых машина может быть обучена без ошибок для всех возможных бинарных маркировок функций классификации.

Пример 2.2

Рассмотрим простое решающее правило в m-мерном пространстве $\mathbf{X}$ входных векторов, описываемое следующим образом:

$$\mathbf{F} : y = \varphi(\mathbf{w}^T\mathbf{x} + b), \tag{2.88}$$

где $\mathbf{x}$ — m-мерный вектор весов; b — порог. Функция активации φ является пороговой, т.е.

$$\varphi(v) = \begin{cases} 1, & v \geq 0, \\ 0, & v < 0. \end{cases}$$

VC-измерение решающего правила, определяемого формулой (2.88), определяется соотношением

$$\text{VCdim}(\mathbf{F}) = m + 1. \tag{2.89}$$

Чтобы проиллюстрировать этот результат, рассмотрим двумерное входное пространство (т.е. $m = 2$), изображенное на рис. 2.24. На рис. 2.24, *а* показаны три точки: $\mathbf{x}_1$, $\mathbf{x}_2$ и $\mathbf{x}_3$, а также три возможных варианта разделения этих точек. На рисунке ясно видно, что точки могут быть разделены тремя линиями. На рис. 2.24, *б* изображены четыре точки: $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ и $\mathbf{x}_4$. Точки $\mathbf{x}_2$ и $\mathbf{x}_3$ относятся к классу 0, а точки $\mathbf{x}_1$ и $\mathbf{x}_4$ — к классу 1. На рисунке видно, что точки $\mathbf{x}_2$ и $\mathbf{x}_3$ нельзя отделить от точек $\mathbf{x}_1$ и $\mathbf{x}_4$ одной линией. Таким образом, VC-измерение решающего правила, описанного формулой (2.88), при $m = 2$ равно 3, что соответствует формуле (2.89). ■

Пример 2.3

Поскольку VC-измерение является мерой емкости множества функций классификации (индикаторов), то можно ожидать, что обучаемая машина с большим числом свободных параметров будет иметь большое VC-измерение, и наоборот. Приведем контрпример, опровергающий это утверждение[13].

Рассмотрим однопараметрическое семейство функций-индикаторов

$$f(x, a) = \text{sgn}(\sin(ax)), a \in \Re,$$

где $\text{sgn}(\cdot)$ — функция вычисления знака аргумента. Предположим, задано некоторое число N, для которого нужно найти N точек, для которых необходимо построить разбиение. Этому требованию удовлетворяет набор функций $f(x, a)$, где

$$x_i = 10^{-i}, i = 1, 2, \ldots, N.$$

Чтобы разделить эти точки данных на два класса, определяемых последовательностью

$$d_1, d_2, \ldots, d_N, d_i \in \{-1, 1\},$$

достаточно выбрать параметр a согласно формуле

$$a = \pi \left(1 + \sum_{i=1}^{N} \frac{(1 - d_i)10^i}{2}\right).$$

Отсюда можно заключить, что VC-измерение семейства функций-индикаторов $f(x, a)$ с единственным свободным параметром a равно бесконечности. ■

Важность VC-измерения и его оценка

VC-измерение является сугубо *комбинаторным понятием* (combinatorial concept) и никак не связано с геометрическим понятием измерения. Оно играет центральную роль в теории статистического обучения, что и будет показано в следующих двух подразделах. VC-измерение важно также и с конструкторской точки зрения. Образно говоря, количество примеров, необходимых для обучения системы данным некоторого класса, строго пропорционально VC-измерению этого класса. Таким образом, VC-измерению стоит уделить первостепенное внимание.

В некоторых случаях VC-измерение определяется свободными параметрами нейронной сети. Однако на практике VC-измерение довольно сложно получить аналитическими методами. Тем не менее границы VC-измерения для нейронной сети часто устанавливаются довольно легко. В этом контексте особый интерес представляют следующие два результата[14].

---

[13] Согласно [165], пример 2.3 впервые был приведен в [1085] благодаря Левину (Levin) и Денкеру (Denker).

[14] Верхний предел порядка $W \log W$ VC-измерения нейронной сети прямого распространения, состоящей из линейных пороговых элементов (персептронов), впервые был получен в [107]. Впоследствии в [692] было показано, что для этого класса нейросетей нижний предел также имеет порядок $W \log W$.

Верхний предел VC-измерения для сигмоидальных нейронных сетей впервые был получен в [698]. Впоследствии авторы работы [586] решили поставленный в [692] следующий вопрос.

1. *Пусть* **N** *— произвольная нейронная сеть прямого распространения, состоящая из нейронов с пороговой функцией активации (Хэвисайда)*

$$\varphi(v) = \begin{cases} 1, & v \geq 0, \\ 0, & v < 0. \end{cases}$$

VC-измерение сети **N** *составляет* $O(W \log W)$, где W *— общее количество свободных параметров сети.*

Этот результат был получен в [107] и [218].

2. *Пусть* **N** *— произвольная многослойная нейронная сеть прямого распространения, состоящая из нейронов с сигмоидальной функцией активации*

$$\varphi = \frac{1}{1 + \exp(-v)}.$$

VC-измерение сети **N** равно $O(W^2)$, W *— общее количество свободных параметров сети.*

Второй результат был получен в [586]. Чтобы получить его, авторы сначала показали, что VC-измерение сетей, состоящее из двух типов нейронов (с линейной и пороговой функциями активации), пропорционально W^2. Это довольно неожиданный результат, так как в примере 2.2 было показано, что VC-измерение чисто линейных нейронных сетей пропорционально W, а VC-измерение нейронных сетей с пороговой функцией активации пропорционально $W \lg W$ (согласно п. 1). Результат, относящийся к нейронным сетям с сигмоидальными функциями активации, был получен с помощью двух аппроксимаций. Во-первых, нейроны с пороговой функцией активации можно аппроксимировать узлами с сигмоидальными функциями и большими синаптическими весами. Во-вторых, линейные нейроны можно аппроксимировать нейронами с сигмоидальными функциями и малыми синаптическими весами.

Здесь важно обратить внимание на то, что многослойные сети прямого распространения имеют *конечное* VC-измерение.

"Действительно ли VC-измерение аналоговой нейронной сети с сигмоидальной функцией активации $\sigma(y) = 1/(1+e^{-y})$ ограничено полиномом с определенным числом программируемых параметров?"

Авторы [586] дали положительный ответ на этот вопрос. Положительный ответ на этот вопрос также был получен в [543], где использовался сложный метод, основанный на дифференциальной топологии. Это было сделано для того, чтобы показать, что VC-измерение в сигмоидальных нейронных сетях, используемых в качестве классификатора образов, ограничено сверху порядком $O(W^4)$. Между этим верхним пределом и нижним пределом, полученным в [586], существует большой разрыв. Однако в [543] было высказано предположение, что данный верхний предел может быть снижен.

Конструктивные, независимые от распределения пределы обобщающей способности обучаемых машин

Теперь следует остановиться на особом типе задач двоичной классификации образов, в которых ожидаемый отклик определяется множеством $d = \{0, 1\}$. Как следствие, функция потерь может принимать одно из двух следующих значений:

$$L(d, F(\mathbf{x}, \mathbf{w})) = \begin{cases} 0, & \text{если } (\mathbf{x},\ \mathbf{w}) = d, \\ 1, & \text{если } F(\mathbf{x},\ \mathbf{w}) \neq d. \end{cases} \tag{2.90}$$

При этих условиях функционалы риска $R(\mathbf{w})$ и эмпирического риска $R_{\text{emp}}(\mathbf{w})$, определяемые формулами (2.72) и (2.74) соответственно, могут иметь следующую интерпретацию.

- Функционал риска $R(\mathbf{w})$ — это *вероятность ошибки классификации* (probability of classification error), обозначаемой $P(\mathbf{w})$.
- Функционал эмпирического риска $R_{\text{emp}}(\mathbf{w})$ — это *ошибка обучения* (training error) (т.е. частота появления ошибок в процессе обучения), обозначаемая $v(\mathbf{w})$.

Согласно *закону больших чисел* (law of large numbers) [380], эмпирическая частота возникновения каких-либо событий почти наверняка сходится к фактической вероятности этих же событий при количестве попыток, стремящемся к бесконечности (подразумевается, что эти попытки независимы и одинаково распределены). В контексте нашего обсуждения этот результат говорит о том, что для любого вектора $\mathbf{w}$, не зависящего от обучающего множества, и для любой точности $\varepsilon > 0$ выполняется следующее условие [1087]:

$$P(|P(\mathbf{w}) - v(\mathbf{w})| > \varepsilon) \to 0 N \to \infty, \tag{2.91}$$

где N — размер множества обучения. Следует заметить, что выполнение условия (2.91) совершенно не означает, что минимизация ошибки обучения $v(\mathbf{w})$ при использовании некоторого правила классификации (т.е. данного вектора весов $\mathbf{w}$) влечет минимизацию вероятности ошибки классификации $P(\mathbf{w})$. Для существенно большого размера N обучающего множества близость между $v(\mathbf{w})$ и $P(\mathbf{w})$ следует из более строгого условия [1087]

$$P(\sup_{\mathbf{w}} |P(\mathbf{w}) - v(\mathbf{w})| > \varepsilon) \to 0 \text{ при } N \to \infty. \tag{2.92}$$

В данном случае речь идет о *равномерной сходимости частоты ошибок обучения к вероятности* того, что $v(\mathbf{w})$=$P(\mathbf{w})$.

Понятие VC-измерения накладывает ограничения на скорость равномерной сходимости. В частности, для множества функций классификации с VC-измерением, равным h, выполняется следующее неравенство [1084], [1087]:

$$P(\sup_{\mathbf{w}} |P(\mathbf{w}) - v(\mathbf{w})| > \varepsilon) < \left(\frac{2eN}{h}\right)^h \exp(-\varepsilon^2 N), \tag{2.93}$$

где N — размер обучающего множества; e — основание натурального логарифма. Для того чтобы достичь равномерной сходимости, требуется обеспечить малое значение правой части неравенства (2.93) для больших значений N. В этом может помочь множитель $\exp(-\varepsilon^2 N)$, так как он экспоненциально убывает с ростом N. Оставшийся множитель $(2eN/h)^h$ представляет собой предел роста функции $\Delta_F(l)$ для семейства функций $\mathbf{F}=\{F(\mathbf{x}, \mathbf{w}); \mathbf{w}\in\mathbf{W}\}$ при $l \geq h \geq 1$. Этот результат описывается *леммой Сауера* (Sauyer's lemma)[15]. Ограничив слишком быстрый рост этой функции, мы обеспечиваем сходимость правой части неравенства к нулю при N, стремящемся к бесконечности. Это требование будет удовлетворено, если VC-измерение h не является бесконечно большим. Другими словами, конечность VC-измерения является необходимым и достаточным условием равномерной сходимости принципа минимизации эмпирического риска. Если входное пространство $\mathbf{X}$ обладает конечной мощностью, то семейство дихотомий $\mathbf{F}$ будет иметь конечное VC-измерение по $\mathbf{X}$. Обратное утверждение не всегда верно.

Пусть α — вероятность события

$$\sup_{\mathbf{w}} |P(\mathbf{w}) - v(\mathbf{w})| \geq \varepsilon.$$

Тогда с вероятностью $(1 - \alpha)$ можно утверждать, что все векторы весовых коэффициентов $\mathbf{w}\in\mathbf{W}$ удовлетворяют следующему неравенству:

$$P(\mathbf{w}) < v(\mathbf{w}) + \varepsilon. \tag{2.94}$$

Используя неравенство (2.93) и определение вероятности α, можно записать:

$$\alpha = \left(\frac{2eN}{h}\right)^h \exp(-\varepsilon^2 N). \tag{2.95}$$

Пусть $\varepsilon_0(N, h, \alpha)$ — некоторое значение ε, удовлетворяющее соотношению (2.95).

---

[15] Лемма Сауера звучит следующим образом [64], [934], [1094].

"Обозначим через $\mathbf{F}$ *ансамбль дихотомий, реализуемых обучаемой машиной. Если* $\mathrm{VCdim}(\boldsymbol{F}) = h$, *где* h — *конечная величина,* $l \geq h \geq 1$, то функция роста $\Delta_F(l)$ *ограничена сверху величиной* $(el/h)^h$, где e — *основание натурального логарифма*".

Тогда можно получить следующий важный результат [1086]:

$$\varepsilon_0(N,h,\alpha)=\sqrt{\frac{h}{N}\left[\log\left(\frac{2N}{h}\right)+1\right]-\frac{1}{N}\log\alpha}. \tag{2.96}$$

Величина $\varepsilon_0(N,n,\alpha)$ называется *доверительным интервалом* (confidence interval). Его значение зависит от размера обучающей выборки N, VC-измерения h и вероятности α.

Предел, описываемый выражением (2.93), при ε=$\varepsilon_0(N,h,\alpha)$ достигается в худшем случае с вероятностью $P(\mathbf{w})$=1/2, но, к сожалению, не для малых значений $P(\mathbf{w})$, которые интересны при решении практических задач. Для малых значений $P(\mathbf{w})$ более полезное ограничение можно получить в результате некоторой модификации неравенства (2.93) [1084], [1087]:

$$P\left(\sup_{\mathbf{w}}\frac{|P(\mathbf{w})-v(\mathbf{w})|}{\sqrt{P(\mathbf{w})}}>\varepsilon\right)<\left(\frac{2eN}{h}\right)^{h}\exp(\frac{-\varepsilon^2N}{4}). \tag{2.97}$$

В литературе представлены различные виды ограничения (2.97), зависящие от конкретной формы неравенства, используемого для их получении. Тем не менее все они имеют сходную форму. Из неравенства (2.97) следует, что с вероятностью $(1-\alpha)$ одновременно для всех $\mathbf{w}\in\mathbf{W}$ выполняется соотношение [1084], [1087]

$$P(\mathbf{w})\le v(\mathbf{w})+\varepsilon_1(N,h,\alpha,v), \tag{2.98}$$

где $\varepsilon_1(N,h,\alpha,v)$ — новый доверительный интервал, определяемый в терминах ранее рассмотренного доверительного интервала $\varepsilon_0(N,h,\alpha)$ следующим образом (см. задачу 2.25):

$$\varepsilon_1(N,h,\alpha,v)=2\varepsilon_0^2(N,h,\alpha)\left(1+\sqrt{1+\frac{v(\mathbf{w})}{\varepsilon_0^2(N,h,\alpha)}}\right). \tag{2.99}$$

Этот доверительный интервал зависит от ошибки обучения $v(\mathbf{w})$. При $v(\mathbf{w})$=0 он принимает упрощенный вид

$$\varepsilon_1(N,h,\alpha,0)=4\varepsilon_0^2(N,h,\alpha). \tag{2.100}$$

Теперь можно подвести итог и определить два ограничения на скорость равномерной сходимости.

1. В общем случае скорость равномерной сходимости удовлетворяет следующему ограничению:

$$P(\mathbf{w}) \leq v(\mathbf{w}) + \varepsilon_1(N, h, \alpha, v),$$

где $\varepsilon_1(N, h, \alpha, v)$ определяется формулой (2.99).

2. При малых (близких к нулю) значениях ошибки обучения $v(\mathbf{w})$ выполняется неравенство

$$P(\mathbf{w}) \leq v(\mathbf{w}) + 4\varepsilon_0^2(N, h, \alpha).$$

Это ограничение является более точным и более пригодным для реальных задач обучения.

3. При больших значениях ошибки обучения $v(\mathbf{w})$, близких к единице, выполняется ограничение

$$P(\mathbf{w}) \leq v(\mathbf{w}) + \varepsilon_0(N, h, \alpha).$$

Минимизация структурного риска

Под *ошибкой обучения* (training error) понимается частота сделанных машиной ошибок в течение сеанса обучения для определенного вектора весов $\mathbf{w}$. Аналогично, под *ошибкой обобщения* (generalization error) понимается частота сделанных машиной ошибок при ее тестировании на не встречавшихся ранее примерах. При этом предполагается, что тестовые данные принадлежат тому же семейству, что и данные обучения. Эти две величины обозначаются как $v_{\text{train}}(\mathbf{w})$ и $v_{\text{gene}}(\mathbf{w})$ соответственно. Заметим, что $v_{\text{train}}(\mathbf{w})$ является *той же* величиной, которая в предыдущем разделе для упрощения записи формул обозначалась $v(\mathbf{w})$. Обозначим символом h VC-измерение семейства функций классификации $\{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathbf{W}\}$ по отношению для пространства входных сигналов $\mathbf{X}$. Тогда в свете теории скорости равномерной сходимости можно утверждать, что с вероятностью $(1 - \alpha)$ для количества примеров обучения $N > h$, одновременно для всех функций классификации $F(\mathbf{x}, \mathbf{w})$ ошибка обобщения $v_{\text{gene}}(\mathbf{w})$ имеет меньшее значение, чем *гарантированный риск* (guaranteed risk), определяемый как сумма пары конкурирующих величин [1084], [1087]

$$v_{\text{guarant}}(\mathbf{w}) = v_{\text{train}}(\mathbf{w}) + \varepsilon_1(N, h, \alpha, v_{\text{train}}), \tag{2.101}$$

где доверительный интервал $\varepsilon_1(N, h, \alpha, v_{\text{train}})$ определяется формулой (2.99). Для фиксированного числа N примеров обучения ошибка обучения монотонно уменьшается при увеличении VC-измерения h, а доверительный интервал монотонно увели-

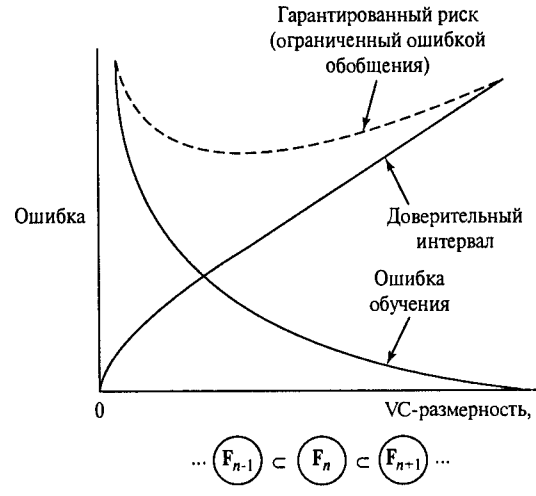


Рис. 2.25. Взаимосвязь между ошибкой обучения, доверительным интервалом и гарантированным риском

чивается. Следовательно, как гарантированный риск, так и ошибка обобщения имеют точку минимума. Общий случай этого утверждения проиллюстрирован на рис. 2.25. До момента достижения точки минимума задача обучения является *переопределенной* (overdetermined) в том смысле, что емкость машины h слишком мала, чтобы вместить весь объем деталей обучения. После прохождения точки минимума задача обучения является *недоопределенной* (underdeterminated), т.е. емкость машины слишком велика для такого объема данных обучения.

Таким образом, при решении задачи обучения с учителем необходимо обеспечить максимальную эффективность обобщения за счет приведения в соответствие емкости машины с доступным количеством данных обучения. *Метод минимизации структурного риска* (method of structural risk minimization) обеспечивает индуктивную процедуру достижения этой цели, в которой VC-измерение обучаемой машины рассматривается как *управляющая* переменная [1084], [1086]. Для большей конкретизации рассмотрим ансамбль классификаторов образов $\{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathbf{W}\}$ и определим вложенную структуру, состоящую из n подобных машин:

$$\mathbf{F}_k = \{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathbf{W}_k\}, k = 1, 2, \ldots, n, \tag{2.102}$$

таких, что (см. рис. 2.25)

$$\mathbf{F}_1 \subset \mathbf{F}_2 \subset \ldots \subset \mathbf{F}_n, \tag{2.103}$$

где символ $\subset$ означает "содержится в". Соответственно VC-измерения отдельных классификаторов образов удовлетворяют следующему условию:

$$h_1 \leq h_2 \leq \ldots \leq h_n. \tag{2.104}$$

Это говорит о том, что VC-измерение каждого из классификаторов конечно. Тогда метод минимизации структурного риска можно изложить следующим образом.

- Минимизируется эмпирический риск (т.е. ошибка обучения) для каждого из классификаторов.
- Определяется классификатор $\mathbf{F}^*$, который имеет наименьший гарантированный риск. Эта конкретная машина обеспечивает наилучший компромисс между ошибкой обучения (т.е. качеством аппроксимации данных обучения) и доверительным интервалом (т.е. сложностью функции аппроксимации), которые конкурируют друг с другом.

Нашей целью является поиск такой нейросетевой структуры, в которой уменьшение VC-измерения достигается за счет минимально возможного увеличения ошибки обучения.

Принцип минимизации структурного риска может быть реализован множеством различных способов. Например, VC-измерение h можно изменять за счет изменения количества скрытых нейронов. В качестве примера рассмотрим ансамбль полносвязных многослойных сетей прямого распространения, в которых количество нейронов в одном из скрытых слоев монотонно возрастает. В соответствии с принципом минимизации структурного риска наилучшей сетью в этом множестве будет та, для которой гарантированный риск будет минимальным.

VC-измерение является основным понятием не только принципа минимизации структурного риска, но и не менее мощной модели обучения, получившей название *вероятностно-корректной в смысле аппроксимации* (probably approximately correct — PAC). Этой моделью, которая описывается в следующем разделе, мы и завершим рассмотрение вероятностных и статистических аспектов обучения.

2.15. Вероятностно-корректная в смысле аппроксимации модель обучения

Вероятностно-корректная в смысле аппроксимации (probably-approximately correct) модель обучения (PAC) была описана в [1075]. Как и следует из ее названия, эта модель представляет собой вероятностный "каркас" (или среду) для изучения процессов обучения и обобщения в системах двоичной классификации. Она тесно связана с принципом обучения с учителем.

Сначала определимся с терминологией, связанной со средой $\mathbf{X}$. Множество из элементов $\mathbf{X}$ называется *понятием* (concept), а любой набор его подмножеств — *классом понятий* (concept class). *Примером* понятия называется любой объект из предметной области, вместе с меткой своего класса. Если пример относится к данному понятию, он называется *положительным примером* (positive example). Если он не относит-

ся к данному понятию, то называется *отрицательным примером* (negative example). Понятие, для которого приводятся примеры, называется *целевым* (target concept). Последовательность данных обучения длины N для понятия c можно определить следующим образом:

$$\mathrm{T} = \{(\mathbf{x}_i,\ c(\mathbf{x}_i))\}_{i=1}^{N}. \tag{2.105}$$

В этой последовательности могут содержаться и повторяющиеся примеры. Примеры $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ выбираются из среды $\mathbf{X}$ случайным образом, в соответствии с некоторым фиксированным, но неизвестным распределением вероятности. В определении (2.105) заслуживают внимания также следующие вопросы.

- Целевое понятие $c(\mathbf{x}_i)$ рассматривается как функция, отображающая $\mathbf{X}$ в множество $\{0, 1\}$. При этом предполагается, что функция $c(\mathbf{x}_i)$ неизвестна.
- Предполагается, что примеры статистически независимы. Это значит, что функция плотности совместной вероятности двух различных примеров $\mathbf{x}_i$ и $\mathbf{x}_j$ равна произведению соответствующих функций плотности вероятности.

В контексте терминологии, которую мы использовали ранее, среда $\mathbf{X}$ соответствует пространству входных сигналов нейронной сети, а целевое понятие — ожидаемому отклику сети.

Набор понятий, порождаемых средой $\mathbf{X}$, называется *пространством понятий* $\mathbf{B}$. Например, пространство понятий может содержать фразы типа "буква А", "буква Б" и т.д. Каждое из этих понятий может быть закодировано различными способами при формировании множеств положительных и отрицательных примеров. В ракурсе обучения с учителем используется другое множество понятий. Обучаемая машина обычно представляет собой множество функций, каждая из которых соответствует определенному состоянию. Например, машина может предназначаться для распознавания "буквы А", "буквы Б" и т.д. Множество всех функций (т.е. понятий), определяемых обучаемой машиной, называется *пространством гипотез* (hypothesis space) $\mathbf{G}$. Это пространство может совпадать или не совпадать с пространством понятий $\mathbf{B}$. С определенной точки зрения пространства понятий и гипотез являются аналогами функции $f(\mathbf{x})$ и аппроксимирующей функции $F(\mathbf{x}, \mathbf{w})$, которыми мы оперировали в предыдущем разделе.

Предположим, что существует некоторое целевое понятие $c(\mathbf{x}) \in \mathbf{B}$, принимающее значения 0 и 1. Требуется обучить этому понятию нейронную сеть при помощи ее настройки на множестве данных T, определенном выражением (2.105). Пусть $g(\mathbf{x}) \in \mathbf{G}$ — гипотеза, соответствующая отображению входа на выход, сформированному в результате проведенного обучения. Одним из способов достижения успеха в обучении является измерение степени близости гипотезы $g(\mathbf{x})$ к целевой концепции $c(\mathbf{x})$. Естественно, всегда существуют ошибки, обеспечивающие различие этих вели-

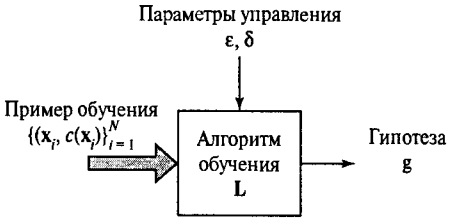


Рис. 2.26. Блочная диаграмма, иллюстрирующая модель обучения PAC

чин. Эти ошибки являются следствием того, что мы пытаемся обучить нейронную сеть некоторой функции на основе ограниченной информации о ней. Вероятность ошибки обучения определяется выражением

$$v_{\text{train}} = P(\mathbf{x} \in \mathbf{X}: g(\mathbf{x}) \neq c(\mathbf{x})). \tag{2.106}$$

Распределение вероятности в этом примере должно быть таким же, как и при формировании примеров. Целью обучения PAC является минимизация значения v_{train}. Предметная область, доступная алгоритму обучения, определяется размером N обучающего множества **T**. Кроме того, алгоритм обучения имеет два следующих параметра управления.

- *Параметр ошибки* (error parameter) $\varepsilon \in (0, 1]$. Этот параметр задает величину ошибки, при которой аппроксимация целевого понятия $c(\mathbf{x})$ гипотезой $g(\mathbf{x})$ считается удовлетворительной.
- *Параметр доверия* (confidence parameter) $\delta \in (0, 1]$. Этот параметр задает степень правдоподобия при построении "хорошей" аппроксимации.

Модель обучения PAC изображена на рис. 2.26.

Теперь можно формально определить модель обучения PAC [551], [1075], [1094].

Пусть **B** *— класс понятий для среды* **X**. *Считается, что класс* **B** *является PAC-обучаемым, если существует алгоритм* **L**, *обладающий следующим свойством. Для любого целевого понятия* $c \in \mathbf{B}$, *для любого распределения вероятности на* **X** *и для всех* $0 < \varepsilon < 1/2$ *и* $0 < \delta < 1/2$ *при использовании алгоритма* **L** *для множества примеров обучения* $\mathbf{T} = \{(\mathbf{x}_i, c(\mathbf{x}_i))\}_{i=1}^N$ *с вероятностью не хуже* $(1 - \delta)$ *результатом алгоритма обучения* **L** *будет гипотеза* g *с ошибкой обучения* $v_{\text{train}} < \varepsilon$. *Эта вероятность получается на любом случайном подмножестве множества* **T** *и при любой внутренней рандомизации, которая может существовать в алгоритме обучения* **L**. *При этом размер обучающего множества* N *должен превышать значение некоторой функции от* δ *и* ε.

Другими словами, если размер N обучающего множества **T** достаточно велик, то существует вероятность, что в результате обучения сети на этом наборе примеров отображение входа на выход, реализуемое сетью, будет "приблизительно корректным". Обратите внимание, что, несмотря на зависимость от ε и δ, количество примеров N не обязательно зависит от целевого понятия c и распределения вероятности в **X**.

Сложность обучающего множества

При использовании теории PAC-обучения на практике возникает интересный вопрос, касающийся *сложности обучающего множества* (sample complexity). Его можно сформулировать следующим образом: сколько случайных примеров нужно предоставить алгоритму обучения, чтобы обеспечить его информацией, достаточной для "изучения" неизвестного понятия c, выбранного из класса понятий $\mathbf{B}$, или насколько большим должно быть обучающее множество $\mathbf{T}$?

Вопрос сложности обучающего множества тесно связан с VC-измерением. Однако, прежде чем продолжить рассмотрение этого вопроса, необходимо определить понятие *согласованности* (consistence). Пусть $\mathrm{T} = \{(\mathbf{x}_i,\ d_i)\}_{i=1}^{N}$ — некоторое множество маркированных примеров, в котором все $\mathbf{x}_i \in \mathbf{X}$ и все $d_i \in (0, 1)$. Тогда понятие c называется согласованным с набором примеров $\mathbf{T}$ (и наоборот, набор $\mathbf{T}$ называется согласованным с c), если для любого $1 \leq i \leq N$ выполняется равенство $c(\mathbf{x}_i) = d_i$ [551]. В свете концепции PAC-обучения критичным является не размер множества вычисляемых нейронной сетью функций отображения входа на выход, а VC-измерение сети. Более точно этот результат можно изложить в виде двух утверждений [64], [134], [1094].

Рассмотрим нейронную сеть с конечным VC-измерением $h \geq 1$.

1. Любой состоятельный алгоритм обучения для этой нейронной сети является алгоритмом обучения PAC.
2. Существует такая константа K, что достаточным размером обучающего множества $\mathbf{T}$ для любого алгоритма является

$$N = \frac{K}{\varepsilon}\left(h \log\left(\frac{1}{\varepsilon}\right) + \log\left(\frac{1}{\delta}\right)\right), \tag{2.107}$$

где ε — параметр ошибки; δ — параметр доверия.

Общность этого результата впечатляет: его можно применить к обучению с учителем, независимо от типа алгоритма обучения и распределения вероятности маркированных примеров. Именно общность этого результата сделала его объектом повышенного внимания в литературе, посвященной нейронным сетям. Сравнение результатов, полученных на основе вычисления VC-измерения, с экспериментальными данными выявило существенные расхождения величин[16]. Это и не удивительно, поскольку та-

[16] В этом примечании будут представлены четыре важных результата, описанных в литературе и посвященных сложности обучающего множества и связанным с ней вопросам обобщения.

Во-первых, в [203] представлено детальное экспериментальное исследование фактических значений сложности обучающего множества, основанное на применении VC-измерения. В частности, в экспериментах предполагалось оценить взаимосвязь между эффективностью обобщения, выполняемого нейронной сетью, и независимым от распределения пессимистическим пределом, полученным на основе теорин статистического обучения Вапника. Ограничение, описанное в [1087], имеет вид $v_{\text{genc}} \geq O\left(\frac{h}{N}\log\left(\frac{h}{N}\right)\right)$, где v_{genc} — ошибка

кое рассогласование отражает *независимость от распределения и пессимистический характер* (distribution-free, worst-case) теоретических оценок. В действительности дела обстоят значительно лучше.

Вычислительная сложность

Еще одним вопросом, который нельзя обойти вниманием при рассмотрении концепции обучения PAC, является вычислительная сложность. Этот вопрос касается вычислительной эффективности алгоритма обучения. Более точно, понятие *вычислительной сложности* (computational complexity) связано с пессимистической оценкой времени, необходимого для обучения нейронной сети (обучаемой машины) на множестве маркированных примеров мощности N.

На практике время работы алгоритма зависит прежде всего от скорости выполнения вычислений. Однако с теоретической точки зрения необходимо дать такое определение времени обучения, которое не будет зависеть от конкретных устройств, используемых для обработки информации. Поэтому время обучения (и соответственно вычислительная сложность) обычно измеряется в терминах количества операций (сложения, умножения и хранения), необходимых для выполнения вычислений.

При оценке вычислительной сложности алгоритма обучения необходимо изначально знать, как она зависит от размерности примеров обучения m (т.е. от размера входного слоя обучаемой нейронной сети). В этом контексте алгоритм считается вычисли-

---

обобщения; h – VC-измерение; N — размер обучающего множества. Результаты, представленные в [203], показали, что средняя эффективность обобщения значительно лучше, чем описанная вышеприведенной формулой.

Во-вторых, в [472] были продолжены исследования, описанные ранее в [203]. Авторы поставили перед собой ту же задачу, но отметили три важных отличия.

- Все эксперименты производились на нейронных сетях с точно известными результатами или очень хорошими пределами, полученными для VC-измерений.
- Рассматривался особый случай алгоритма обучения.
  Эксперименты были основаны на реальных данных.
- Хотя в этой работе были получены более ценные с практической точки зрения оценки сложности обучающего множества, работа имела ряд существенных теоретических изъянов, от которых нужно было избавиться.

В 1989 году вышла в свет работа, посвященная оцениванию размера N обучающего множества, необходимого для обучения однослойной сети прямого распространения с линейными пороговыми нейронами с обеспечением хорошего качества обобщенния [107]. Предполагалось, что обучающие примеры выбираются случайно с произвольным распределением вероятности, а тестовые примеры, используемые для оценки эффективностн обобщения, имеют такое же распределение. Согласно [107], сеть почти наверняка обеспечивает обобщение, если выполняются следующие условия.

1. Количество ошибок на обучающем множестве не превышает величины $\varepsilon/2$.
2. Количество примеров, используемых для обучения, удовлетворяет соотношению $N \geq O\left(\frac{W}{\varepsilon}\log\left(\frac{W}{\varepsilon}\right)\right)$, где W — число синаптических весов в сети. Это неравенство описывает независимое от распределения пессимистическое ограничение, накладываемое на размер N. Но и здесь может наблюдаться большое численное расхождение между фактически необходимым размером обучающего множества и оценкой, обеспечиваемой этим неравенством.

Наконец, в 1997 году в [97] был поднят вопрос о том, что в задачах классификации образов с помощью больших нейронных сетей часто оказывается, что для успешного обучения сеть способна обойтись гораздо меньшим количеством примеров, чем количество синаптических весов в ней (что утверждалось в [203]). В [97] было показано, что в тех задачах, где нейронная сеть при небольшом количестве синаптических весов хорошо выполняет обобщение, эффективность этого обобщения определяется не количеством весов, а их величиной.

тельно *эффективным* (efficient), если время его работы пропорционально $O(m^r)$, где $r \geq 1$. В этом случае говорят, что время обучения полиномиально зависит от m (polynomically with m), а сам алгоритм называется *алгоритмом с полиномиальным временем выполнения* (polynomial time algorithm). Задачи обучения, основанные на алгоритмах с полиномиальным временем выполнения, обычно считаются "простыми" [64].

Еще одним параметром, который требует особого внимания, является параметр ошибки ε. Если речь идет о сложности обучающего множества, параметр ошибки ε является фиксированным, но произвольным; а при оценке вычислительной сложности алгоритма обучения необходимо знать, как она зависит от этого параметра. Интуитивно понятно, что при уменьшении параметра ε задача обучения усложняется. Отсюда следует, что со временем должно быть достигнуто некоторое состояние, которое обеспечит вероятностно-корректный в смысле аппроксимации выход. Для обеспечения эффективности вычислений соответствующее состояние должно достигаться за полиномиальное время по $1/\varepsilon$.

Объединив эти рассуждения, можно сформулировать следующее формальное утверждение [64].

Алгоритм обучения является вычислительно эффективным по параметру ошибки ε, *размерности примеров обучения* m *и размеру обучающего множества* N, *если время его выполнения является полиномиальным по* N, *и существует такое значение* $N_0(\delta, \varepsilon)$, *достаточное для PAC-обучения, при котором алгоритм является полиномиальным по* m *и* ε^{-1}.

2.16. Резюме и обсуждение

В этой главе мы обсудили некоторые важные вопросы, затрагивающие различные аспекты процесса обучения в контексте нейронных сетей. Таким образом, были заложены основы для понимания материала оставшейся части книги. При создании нейронных сетей используются пять правил обучения: *обучение на основе коррекции ошибок* (error-correction learning), *обучение на основе памяти* (memory-based learning), *обучение Хебба* (Hebb's learning), *конкурентное обучение* (competitive learning) и *обучение Больцмана* (Boltzmann learning). Некоторые из этих алгоритмов требуют наличия учителя, а некоторые — нет. Важно то, что эти правила позволяют значительно продвинуться за рамки линейных адаптивных фильтров, как в смысле расширения возможностей, так и в смысле повышения универсальности.

При исследовании методов обучения с учителем ключевым является понятие "учителя", призванного вносить уточняющие коррективы в выходной сигнал нейронной сети при возникновении ошибок (в обучении на основе коррекции ошибок) или "загонять" свободные входные и выходные элементы сети в рамки среды (при обучении Больцмана). Ни одна из этих моделей принципиально не применима к биологическим организмам, которые не содержат двусторонних нервных соединений, так необходи-

мых для обратного распространения сигналов при коррекции ошибок (в многослойных сетях), а также не поддерживают механизма корректировки поведения извне. Тем не менее обучение с учителем утвердилось в качестве мощной парадигмы обучения искусственных нейронных сетей. В этом можно убедиться при ознакомлении с главами 3–7 настоящей книги.

В отличие от метода обратного распространения правила обучения без учителя на основе самоорганизации (такие как правило Хебба или конкурентное обучение) основаны на принципах нейробиологии. Однако для более четкого понимания этого типа обучения необходимо ознакомиться с основами *теории информации Шеннона*. При этом нельзя обойти вниманием *принцип полного количества информации* (mutual information principle) или *Инфомакса* (Infomax) [653], [654], который обеспечивает математический формализм обработки информации в самоорганизующихся нейронных сетях по аналогии с передачей информации по каналам связи. Принцип Инфомакса и его вариации подробно изложены в главе 10.

Обсуждение методов обучения было бы не полным, если бы мы не остановились на *модели селективного обучения Дарвина* (Darwinian selective learning model) [275], [876]. *Отбор* (selection) является важным биологическим принципом, который применяется в теории эволюции и развития. Он лежит в основе работы иммунной системы [276] — наиболее понятной биологической системы распознавания. Модель селективного обучения Дарвина основана на *теории отбора групп нейронов* (theory of neural group selection). В ней предполагается, что нервная система работает по принципу естественного отбора, с тем отличием от эволюционного процесса, что все действие разворачивается в мозге и только на протяжении жизни организма. Согласно этой теории, основной операционной единицей нервной системы является отнюдь не отдельный нейрон, а локальная группа жестко связанных клеток. Принадлежность отдельных нейронов к различным группам определяется изменениями синаптических весов. Локальная конкуренция и кооперация клеток обеспечивают локальный порядок сети. Коллекция групп нейронов называется *полем* или *репертуаром* (repertoire). Ввиду случайной природы развития нейронной сети группы одного поля хорошо подходят для описания перекрывающихся, но подобных входных образов. За каждый входной образ отвечает одна или несколько групп некоторого поля. Тем самым обеспечивается отклик на неизвестные входные образы, который может оказаться очень важным. Селективное обучение Дарвина отличается от всех других алгоритмов, обычно используемых в нейронных сетях, поскольку в структуре нейронной сети предполагается наличие множества подсетей, из которых в процессе обучения отбираются только те, выходной сигнал которых ближе всего к ожидаемому отклику.

Это обсуждение можно завершить несколькими замечаниями, касающимися статистического и вероятностного аспектов обучения. VC-измерение зарекомендовало себя как основной параметр теории статистического обучения. На нем основаны принцип минимизации структурного риска и вероятностно-корректная в смысле аппроксимации модель обучения (PAC). VC-измерение является составной частью теории так называемых машин опорных векторов, которые рассматриваются в главе 6.

По мере ознакомления с материалом настоящей книги читателю еще не раз придется возвращаться к настоящей главе, посвященной основам процессов обучения.

Задачи

Правила обучения

2.1. Дельта-правило, описанное в формуле (2.3), и правило Хебба, описываемое соотношением (2.9), представляют собой два различных метода обучения. Укажите различия между этими двумя правилами.

2.2. Правило обучения на основе коррекции ошибок можно реализовать с помощью запрета вычитания желаемого отклика (целевого значения) из фактического выходного сигнала с последующим применением антихеббовского правила [748]. Прокомментируйте эту интерпретацию обучения на основе коррекции ошибок.

2.3. На рис. 2.27 показано двумерное множество точек. Часть этих точек принадлежит классу $\mathbf{C}_1$, остальные — классу $\mathbf{C}_2$. Постройте границу решений, применив правило ближайшего соседа.

2.4. Рассмотрим группу людей, коллективное мнение которых по некоторому интересующему нас вопросу определяется как взвешенное среднее мнений отдельных членов группы. Предположим, что при изменении мнения члена группы в сторону коллективной точки зрения оно приобретает больший вес. С другой стороны, если отдельные члены группы встают в открытую оппозицию коллективной точке зрения, их мнение теряет вес. Эта форма взвешивания эквивалентна управлению с положительной обратной связью, конечным результатом которого является достижение общего консенсуса среди членов группы [653].

Проведите аналогию между этой ситуацией и постулатом обучения Хебба.

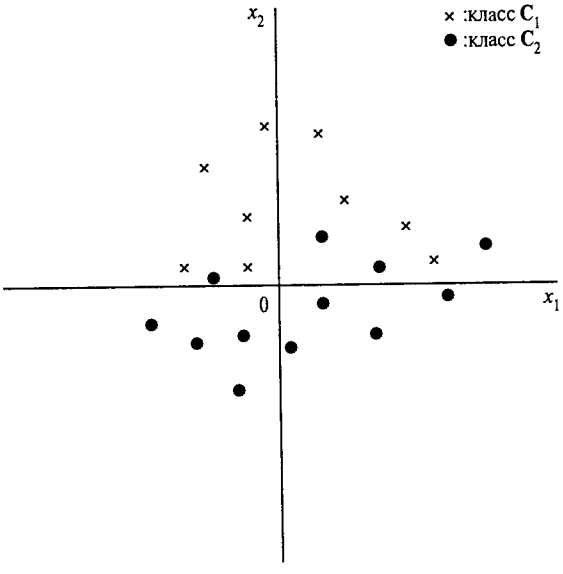


Рис. 2.27. Пример обучающего множества

2.5. Обобщенная форма правила Хебба описывается соотношением

$$\Delta w_{kj}(n) = \alpha\, F(y_k(n))G(x_j(n)) - \beta\, w_{kj}(n)F(y_k(n)),$$

где $x_j(n)$ и $y_k(n)$ — предсинаптический и постсинаптический сигналы соответственно; $F(\cdot)$ и $G(\cdot)$ — некоторые функции своих аргументов; $\Delta w_{kj}(n)$ — изменение синаптического веса w_{kj} в момент времени n, вызванное сигналами$x_j(n)$ и $y_k(n)$. Найдите точку равновесия и максимальное значение спуска, определяемое этим правилом.

2.6. Входной сигнал с единичной амплитудой многократно передается через синаптическую связь, начальное значение которой также равно единице. Вычислите изменение значений синаптического веса со временем с помощью следующих правил:

а) простой формы правила Хебба (2.9), в предположении, что параметр скорости обучения η =0,1;

б) правила ковариации (2.10) для значений предсинаптической $\bar{x} = 0$ и постсинаптической $\bar{y} = 1.0$ активности.

2.7. Синапс Хебба, описываемый соотношением (2.9), предполагает использование положительной обратной связи. Докажите истинность этого утверждения.

2.8. Рассмотрим гипотезу ковариации для обучения на основе самоорганизации, описываемого выражением (2.10). В предположении эргодичности системы (когда среднее по времени может быть заменено средним по множеству) покажите, что ожидаемое значение Δw_{kj} в (2.10) можно представить соотношением

$$E[\Delta w_{kj}] = \eta\,(E[y_k x_j] - \bar{y}\bar{x}). \tag{2.108}$$

Как можно интерпретировать этот результат?

2.9. Согласно [656], постулат Хебба формулируется следующим образом:

$$\Delta w_{ki} = \eta\,(y_k - y_o)(x_i - x_o) + a_1,$$

где x_i и y_k — предсинаптический и постсинаптический сигналы соответственно; a_1, η, x_o — константы. Предположим, что нейрон k — линейный, т.е.

$$y_k = \sum_j w_{kj} x_j + a_2,$$

где a_2 — еще одна константа. Предположим, что все входные сигналы имеют одинаковое распределение, т.е. $E[x_i] = E[x_j] = \mu$. Пусть $\mathbf{C}$ — матрица ковариации входного сигнала, элемент ij которой определяется соотношением

$$C_{ij} = E[(x_i - \mu)(x_j - \mu)].$$

Найдите $E[\Delta w_{kj}]$.

2.10. Напишите выражение для описания выходного сигнала y_j нейрона j в сети, показанной на рис. 2.4. При этом можно использовать следующие переменные:

x_i — i-й входной сигнал;
w_{ij} — синаптический вес связи входа i с нейроном j;
c_{kj} — вес латеральной связи нейрона k с нейроном j;
v_i — индуцированное локальное поле нейрона j;
$y_j = \varphi(v_i)$.

Какому условию должен удовлетворять нейрон j, чтобы выйти победителем?

2.11. Решите задачу 2.10 при условии, что каждый выходной нейрон имеет обратную связь с самим собой.

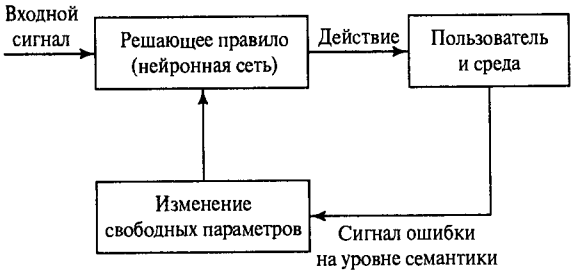


Рис. 2.28. Блочная диаграмма адаптивной системы изучения естественного языка

2.12. Принцип латерального торможения сводится к следующему: "близкие нейроны возбуждаются, а дальние — тормозятся". Его можно промоделировать с помощью разности между двумя гауссовыми кривыми. Эти две кривые имеют общую область определения, но положительная кривая возбуждения имеет более крутой и высокий пик, чем отрицательная кривая торможения. Это значит, что модель такого соединения можно описать соотношением

$$W(x) = \frac{1}{\sqrt{2\pi}\sigma_e} e^{-x^2/\,2\sigma_e^2} - \frac{1}{\sqrt{2\pi}\sigma_i} e^{-x^2/\,2\sigma_i^2},$$

где x — расстояние до нейрона, отвечающего за латеральное торможение. Шаблон $W(x)$ используется для сканирования страницы, одна половина которой белая, а другая — черная. Граница между двумя половинками перпендикулярна оси x.

Постройте график изменения выхода процесса сканирования для σ_e =5; σ_i =8 и σ_e = 1, σ_i =2.

Парадигмы обучения

2.13. На рис. 2.28 показана блочная диаграмма *адаптивной системы изучения естественного языка* (adaptive language-acquisition system) [371]. Синаптические связи в нейронной сети, входящей в состав этой системы, усиливаются и ослабляются в зависимости от значения обратной связи, полученной в результате реакции системы на входное возбуждение. Этот принцип можно рассматривать как пример обучения с подкреплением. Обоснуйте правильность этого утверждения.

2.14. К какой из двух парадигм (обучение с учителем и без него) можно отнести следующие алгоритмы:

а) правило ближайшего соседа;

б) правило k ближайших соседей;

в) обучение Хебба;

г) правило обучения Больцмана.

Обоснуйте свой ответ.

2.15. Обучение без учителя может быть реализовано в кумулятивном и интерактивном режимах. Опишите физический смысл этих двух возможностей.

2.16. Рассмотрим сложности, с которыми сталкивается обучаемая машина при назначении коэффициентов доверия результату игры в шахматы (победа, поражение, ничья). Объясните понятия временной и структурной задач присваивания коэффициентов доверия в контексте этой игры.

2.17. Задачу обучения с учителем можно рассматривать как задачу обучения с подкреплением, использующую в качестве подкрепления некоторую меру близости фактического отклика системы с желаемым. Опишите взаимосвязь между обучением с учителем и обучением с подкреплением.

Память

2.18. Рассмотрим следующее ортонормальное множество ключевых образов, применяемых для построения корреляционной матрицы памяти:

$$\begin{aligned}\mathbf{x}_1 &= [1,0,0,0]^T,\\ \mathbf{x}_2 &= [0,1,0,0]^T,\\ \mathbf{x}_3 &= [0,0,1,0]^T\end{aligned}$$

и соответствующее множество запомненных образов:

$$\begin{aligned}\mathbf{y}_1 &= [5,1,0]^T,\\ \mathbf{y}_2 &= [-2,1,6]^T,\\ \mathbf{y}_3 &= [-2,4,3]^T.\end{aligned}$$

а) Вычислите матрицу памяти $\mathbf{M}$.

б) Покажите, что эта память обладает хорошими ассоциативными способностями.

2.19. Вернемся к предыдущей задаче построения корреляционной матрицы памяти. Предположим, что применяемый к этой памяти стимул представляет собой зашумленную версию ключевого образа $\mathbf{x}_1$ вида

$$\mathbf{x} = [0,8; -0,15; 0,15; -0,20]^T.$$

а) Вычислите отклик памяти $\mathbf{y}$.

б) Покажите, что отклик $\mathbf{y}$ наиболее близок к запомненному образу $\mathbf{y}_1$ в Евклидовом смысле.

2.20. Автоассоциативная память обучается на следующих ключевых векторах:

$$\begin{aligned}\mathbf{x}_1 &= \tfrac{1}{4}[-2, -3, \sqrt{3}]^T,\\ \mathbf{x}_2 &= \tfrac{1}{4}[2, -2, -\sqrt{8}]^T,\\ \mathbf{x}_3 &= \tfrac{1}{4}[3, -1, \sqrt{6}]^T.\end{aligned}$$

а) Вычислите углы между этими векторами. Насколько эти векторы близки к ортогональным?

б) Используя обобщение правила Хебба (т.е. правило внешнего произведения), вычислите матрицу памяти для этой сети. Исследуйте, насколько эта матрица близка к идеальной автоассоциативной памяти.

в) В систему ассоциативной памяти подается "замаскированная" версия ключевого вектора $\mathbf{x}_1$:

$$\mathbf{x} = [0, -3, \sqrt{3}]^T. \tag{2.109}$$

Вычислите реакцию памяти и сравните ее с ожидаемым откликом $\mathbf{x}_1$.

Адаптация

2.21. На рис. 2.29 представлена блочная диаграмма некоторой адаптивной системы. Входной сигнал *модели прогнозирования* определяется предыдущими значениями процесса, а именно:

$$\mathbf{x}(n-1) = [x(n-1), x(n-2), \ldots, x(n-m)]^T.$$

Выход модели $\hat{x}(n)$ представляет собой *оценку* текущего значения $x(n)$ процесса. Компаратор вычисляет сигнал ошибки

$$e(n) = x(n) - \hat{x}(n),$$

который, в свою очередь, корректирует настраиваемые параметры модели. Выходной сигнал также направляется на следующий уровень нейросетевой обработки для интерпретации. При многократном повторении этой операции качество обработки информации системой значительно возрастает [719].

Дополните рис. 2 29 информацией об уровне обработки сигнала.

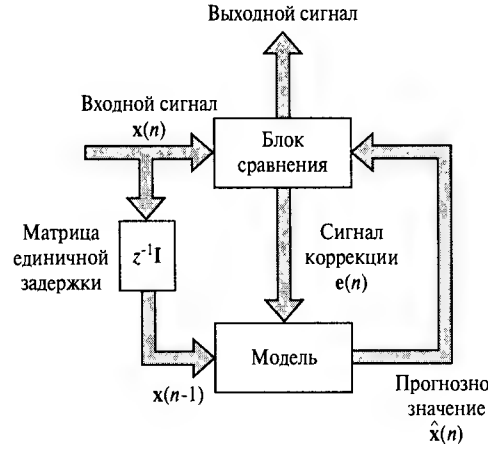


Рис. 2.29. Блочная диаграмма адаптивной системы

Статистическая теория обучения

2.22. Выполняя процедуру, использованную для вывода соотношения (2.62) из (2.61), выведите формулу (2.66), определяющую среднюю по ансамблю функцию $L_{av}(f(\mathbf{x}), F(\mathbf{x}, \mathbf{T}))$.

2.23. В этой задаче необходимо вычислить VC-измерение прямоугольной области, ориентированной вдоль одной из осей на плоскости. Покажите, что VC-измерение этой области равно четырем. Рассмотрите следующие варианты.

а) На плоскости имеются четыре точки, а дихотомия реализована прямоугольниками, ориентированными по осям.

б) На плоскости имеются четыре точки, для которых не существует реализуемой дихотомии в виде прямоугольников, ориентированных по осям.

в) На плоскости имеются пять точек, для которых не существует дихотомии, реализуемой в виде прямоугольников, ориентированных по осям.

2.24. Рассмотрим линейный двоичный классификатор образов, входной вектор $\mathbf{x}$ которого имеет размерность m. Первый элемент вектора $\mathbf{x}$ является константой, равной единице, так что соответствующий вес классификатора описывает пороговое значение (bias). Каково VC-измерение этого классификатора по входному пространству?

2.25. Неравенство (2.97) определяет ограничение на скорость равномерной сходимости, лежащей в основе принципа минимизации эмпирического риска.

а) Докажите истинность формулы (2.98) при условии выполнения неравенства (2.97).

б) Выведите формулу (2.99), определяющую доверительный интервал ε_1.

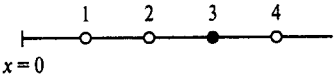


Рис. 2.30. Множество точек для классификации

2.26. Продолжая пример 2.3, покажите, что четыре равноудаленные точки на рис. 2.30 не могут быть разделены однопараметрическим семейством индикаторных функций $f(x, a)$, где $a \in \Re$.

2.27. Опишите взаимосвязь дилеммы смещения/дисперсии и принципа минимизации структурного риска в контексте нелинейной регрессии.

2.28. а) Алгоритм, используемый для обучения многослойной сети прямого распространения с сигмоидальной активационной функцией, является PAC-обучаемым. Обоснуйте истинность этого утверждения.

б) Выполняется ли аналогичное утверждение для произвольной нейронной сети с пороговой функцией активации? Обоснуйте свой ответ.

ГЛАВА 3

Однослойный персептрон

3.1. Введение

Можно выделить нескольких исследователей, которые внесли действительно выдающийся вклад в развитие теории искусственных нейронных сетей в годы становления этой проблемной области (1943–1958).

- Мак-Каллок (McCalloch) и Питц (Pitt) в 1943 году впервые представили идею использования нейронных сетей в качестве вычислительных машин [714].
- Хебб (Hebb) в 1949 году ввел первое правило самоорганизующегося обучения [445].
- Розенблатт (Rosenblatt) в 1958 году ввел понятие персептрона как первой модели обучения с учителем [902].

Влияние работы Мак-Каллока и Питца на развитие нейронных сетей уже обсуждалось в главе 1, а идея обучения Хебба вкратце рассматривалась в главе 2. В этой главе речь пойдет о *персептроне* Розенблатта (Rozenblatt perceptron).

Персептрон представляет собой простейшую форму нейронной сети, предназначенную для классификации *линейно-разделимых* (linearly separable) сигналов (т.е. образы можно разделить некоторой гиперплоскостью). Персептрон состоит из одного нейрона с настраиваемыми синаптическими весами и порогом. Первый алгоритм настройки свободных параметров для такой нейронной сети был создан Розенблаттом [899], [902] для персептронной модели мозга[1]. Розенблатт доказал, что если образы (векторы), используемые для обучения персептрона, выбраны из двух линейно-разделимых классов, то алгоритм персептрона сходится и формирует поверхность решений в форме гиперплоскости, разделяющей эти два класса. Доказательство схо-

[1] В исходной версии персептрона, согласно Розенблатту [899], содержались три типа элементов: сенсориые, ассоциативные и реактивные. Веса связей сенсорных элементов с ассоциативными были фиксированными, а веса связей ассоциативных элемеитов с реактивными — переменными. Ассоциативные элементы выступали в роли препроцессоров, предназначенных для извлечения модели из данных среды. Что касается переменных весов, то функционирование исходного персептрона Розенблатта в сущности соответствует случаю простого реактивного элемента (т.е. одного нейрона).

димости этого алгоритма получило название *теоремы о сходимости персептрона* (perceptron convergence theorem). Персептрон, построенный на одном нейроне, ограничен выполнением задачи разделения только двух классов (гипотез). Увеличивая размерность выходного (вычислительного) слоя персептрона и включая в него несколько нейронов, можно решать задачи классификации на большее число классов. Важно заметить, что при описании теории персептрона достаточно рассмотреть случай сети с единственным нейроном. Обобщение этой теории на класс систем, содержащих несколько нейронов, довольно тривиально.

Единичный нейрон также составляет основу *адаптивного фильтра* (adaptive filter) — функционального блока, являющегося основным для предметной области *обработки сигналов* (signal processing). Развитие теории адаптивной фильтрации было вызвано новаторской работой, открывшей миру так называемый *алгоритм минимизации среднеквадратической ошибки LMS* (least-mean-square algorithm), известный также под названием *дельта-правила* (delta rule) [1141]. Несмотря на свою простоту, алгоритм продемонстрировал высокую эффективность. В самом рабочем названии задачи *линейной адаптивной фильтрации* (linear adaptive filtering) подразумевается, что нейрон работает в линейном режиме. Адаптивные фильтры могут успешно применяться в таких разноплановых областях, как управление антеннами, системы связи и управления, радары, сейсмология и биомедицинская инженерия [434], [435], [1144].

Алгоритм LMS и персептрон тесно связаны между собой, поэтому имеет смысл рассмотреть их вместе в одной главе.

Структура главы

Настоящая глава состоит из двух частей. В первой части (разделы 3.2–3.7) рассматриваются линейные адаптивные фильтры и алгоритм LMS. Вторая часть (разделы 3.8–3.10) посвящена персептрону Розенблатта. Такой порядок изложения сохраняет историческую хронологию появления этих понятий.

В разделе 3.2 описывается задача адаптивной фильтрации. В разделе 3.3 рассматриваются три метода безусловной оптимизации: методы наискорейшего спуска, Ньютона и Ньютона–Гаусса. Эти методы имеют самое прямое отношение к изучению адаптивных фильтров. В разделе 3.4 речь пойдет о линейном фильтре, основанном на методе наименьших квадратов, который асимптотически сходится к фильтру Винера по мере увеличения выборки данных. Фильтр Винера реализует идеальный, в смысле производительности, линейный адаптивный фильтр, работающий в стационарной среде. В разделе 3.5 рассматривается алгоритм LMS и обсуждаются его возможности и ограничения. В разделе 3.6 раскрывается идея кривых обучения, широко используемых для оценки производительности адаптивных фильтров. В разделе 3.7 обсуждается принцип имитации отжига для алгоритма LMS.

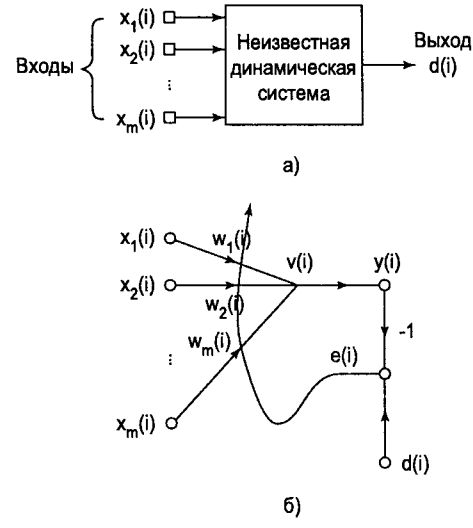


Рис. 3.1. Неизвестная динамическая система (а). Граф прохождения сигнала в адаптивной модели системы (б)

Раздел 3.8 посвящен персептрону Розенблатта — в нем вводится ряд базовых понятий, связанных с его работой. В разделе 3.9 описывается алгоритм настройки вектора синаптических весов персептрона при решении задачи классификации образов из двух линейно-разделимых классов и обосновывается его сходимость. В разделе 3.10 анализируется взаимосвязь между персептроном и байесовским классификатором для гауссовой среды.

Глава завершается в разделе 3.11, в котором подводятся итоги.

3.2. Задача адаптивной фильтрации

Рассмотрим *динамическую систему* (dynamical system), математические характеристики которой *неизвестны*. В нашем распоряжении имеется только набор маркированных данных входного и выходного сигналов, генерируемых системой в равномерные дискретные интервалы времени. В частности, если m входных узлов генерируют m-мерное воздействие $\mathbf{x}(i)$, в ответ система формирует скалярный выходной сигнал $d(i)$, где $i = 1, 2, \ldots, n$ (рис. 3.1). Таким образом, внешнее поведение системы описывается следующим множеством данных:

$$\mathbf{T} : \{\mathbf{x}(i), d(i); i = 1, 2, \ldots, n, \ldots\}, \tag{3.1}$$

где

$$\mathbf{x}(i) = [x_1(i), x_2(i), \ldots, x_m(i)]^T.$$

Множество $\mathbf{T}$ содержит примеры, одинаково распределенные согласно некоторому вероятностному закону. Размерность входного вектора $\mathbf{x}(i)$ называют *размерностью входного пространства* (dimensionality of the input space).

Входной сигнал $\mathbf{x}(i)$ может быть представлен двумя диаметрально противоположными способами — в пространстве и во времени.

- Если m элементов сигнала $\mathbf{x}(i)$ зарождаются в различных точках пространства, то входной сигнал можно считать *моментальным снимком данных* (snapshot of data).
- Если m элементов сигнала $\mathbf{x}(i)$ представляют собой текущее и $(m-1)$ предыдущее значения возбуждения, *снятые через равномерные промежутки времени* (uniformly spaced in time), говорят, что входной сигнал снимается во временной области.

Требуется построить модель выходного сигнала неизвестной динамической системы с несколькими входами и одним выходом на основе одного нейрона. Нейронная модель работает под управлением некоторого *алгоритма*, который обеспечивает настройку синаптических весов нейрона. При этом принимаются следующие соглашения.

- Алгоритм начинает работу с *произвольных* (arbitrary) значений синаптических весов.
- Настройка синаптических весов в ответ на статистические вариации поведения системы выполняется *непрерывно* (continuous) (т.е. время встроено в структуру самого алгоритма).
- Вычисление корректирующих значений синаптических весов выполняется через равномерные промежутки времени.

Нейронная модель, описываемая таким образом, получила название *адаптивного фильтра* (adaptive filter). Несмотря на то что из самого названия однозначно следует назначение системы — идентификация систем, — характеристики адаптивных фильтров выводят нас далеко за рамки этого применения.

На рис. 3.1, *б* показан граф прохождения сигнала в адаптивном фильтре. Его работа включает два последовательных процесса.

1. *Процесс фильтрации* (filtering process), предполагающий вычисление двух сигналов.
   - Выходной сигнал, обозначаемый как $y(i)$ и генерируемый в ответ на вектор входного воздействия $\mathbf{x}(i)$ с компонентами $x_1(i), x_2(i), \ldots, x_m(i)$.
   - Сигнал ошибки, обозначаемый как $e(i)$ и вычисляемый как отклонение выходного сигнала $y(i)$ от выходного сигнала реальной системы $d(i)$, который еще называют *целевым сигналом* (target signal) или *ожидаемым откликом* (desired response).

2. *Процесс адаптации* (adaptive process), включающий автоматическую подстройку синаптических весов нейрона на основе сигнала ошибки $e(i)$.

Комбинация этих двух процессов называется *контуром с обратной связью* (feedback loop) нейрона.

Учитывая линейность нейрона, выходной сигнал $y(i)$ совпадает с индуцированным локальным полем $v(i)$:

$$y(i) = v(i) = \sum_{k=1}^{m} w_k(i)x_k(i), \tag{3.2}$$

где $w_1(i), w_2(i), \ldots, w_m(i)$ — m синаптических весов нейрона, измеренных в момент времени i. В матричной форме выходной сигнал $y(i)$ можно представить как скалярное произведение векторов $\mathbf{w}(i)$ и $\mathbf{x}(i)$:

$$y(i) = \mathbf{x}^T(i)\mathbf{w}(i), \tag{3.3}$$

где

$$\mathbf{w}(i) = [w_1(i), w_2(i), \ldots, w_m(i)]^T.$$

Заметим, что индексация синаптических весов несколько упрощена и не содержит индекса, указывающего на конкретный нейрон. Это обусловлено тем, что мы рассматриваем случай одного нейрона. Это соглашение распространяется на всю настоящую главу. Разность значения выходного сигнала нейрона $y(i)$ и фактического выходного сигнала системы $d(i)$ составляет сигнал ошибки $e(i)$:

$$e(i) = y(i) - d(i). \tag{3.4}$$

Алгоритм применения сигнала ошибки для коррекции синаптических весов нейрона определяется функцией стоимости, используемой конкретным методом адаптивной фильтрации. Этот вопрос тесно связан с задачей оптимизации. Таким образом, уместно предложить обзор некоторых методов оптимизации, которые можно применять не только к линейным адаптивным фильтрам, но и к нейронным сетям в целом.

3.3. Методы безусловной оптимизации

Рассмотрим *непрерывно дифференцируемую* функцию стоимости $\mathbf{E}(\mathbf{w})$, зависящую от некоторого неизвестного вектора $\mathbf{w}$. Функция $\mathbf{E}(\mathbf{w})$ отображает элементы вектора $\mathbf{w}$ в пространство действительных чисел и является мерой оптимальности выбранного для алгоритма адаптивной фильтрации вектора $\mathbf{w}$. Требуется отыскать такое решение $\mathbf{w}^*$, что

$$\mathbf{E}(\mathbf{w}*) \leq \mathbf{E}(\mathbf{w}), \tag{3.5}$$

т.е. необходимо решить *задачу безусловной оптимизации* (unconstrained optimization problem), которая формулируется следующим образом.

Минимизировать функцию стоимости $\mathbf{E}(\mathbf{w})$ *по отношению к вектору весов* $\mathbf{w}$

$$E(\mathbf{w}) \rightarrow \min. \tag{3.6}$$

Необходимым условием оптимальности является следующее:

$$\nabla E(\mathbf{w}*) = 0, \tag{3.7}$$

где ∇ — *оператор градиента*

$$\nabla = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \ldots, \frac{\partial}{\partial w_m}\right]^T, \tag{3.8}$$

а $\nabla\mathbf{E}(\mathbf{w})$ — *вектор градиента* функции стоимости

$$\nabla\mathbf{E}(\mathbf{w}) = \left[\frac{\partial\mathbf{E}}{\partial w_1}, \frac{\partial\mathbf{E}}{\partial w_2}, \ldots, \frac{\partial\mathbf{E}}{\partial w_m}\right]^T. \tag{3.9}$$

Из всех методов безусловной оптимизации для создания адаптивных фильтров лучше всего подходят алгоритмы *последовательного спуска* (iterative descent).

Начиная с исходного значения $\mathbf{w}(0)$ *генерируется последовательность векторов весовых коэффициентов* $\mathbf{w}(1)$, $\mathbf{w}(2)$, ..., *таких, что с каждой итерацией алгоритма значение функции стоимости уменьшается:*

$$\mathbf{E}(\mathbf{w}(n+1)) < \mathbf{E}(\mathbf{w}(n)), \tag{3.10}$$

где $\mathbf{w}(n)$ — *предыдущее значение вектора весов;* $\mathbf{w}(n+1)$ — *последующее.*

Можно надеяться, что такой алгоритм сойдется к оптимальному решению $\mathbf{w}^*$. Слово "надеяться" здесь появилось не случайно — остается вероятность, что алгоритм будет расходиться (т.е. станет неустойчивым), если не будут выполнены определенные условия.

В этом разделе описываются три метода безусловной оптимизации, основанные на той или иной реализации идеи итеративного спуска [124].

Метод наискорейшего спуска

В методе наискорейшего спуска корректировка векторов весов выполняется в направлении максимального уменьшения функции стоимости, т.е. в направлении, противоположном *вектору градиента* (gradient vector) $\nabla \mathbf{E}(\mathbf{w})$. Для удобства можно принять следующее обозначение градиента:

$$\mathbf{g} = \nabla E(\mathbf{w}). \tag{3.11}$$

Формально алгоритм наискорейшего спуска можно записать в следующем виде:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta \mathbf{g}(n), \tag{3.12}$$

где η — положительная константа, называемая *параметром скорости обучения* (learning-rate parameter); $\mathbf{g}(n)$ — вектор градиента, вычисленный в точке $\mathbf{w}(n)$. Переходя от n-й итерации к $n+1$-й, алгоритм выполняет *коррекцию* (correction) весовых коэффициентов:

$$\Delta \mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) = -\eta \mathbf{g}(n). \tag{3.13}$$

При внимательном рассмотрении видно, что уравнение (3.13) является формальной записью описанного в главе 2 правила коррекции ошибок.

Чтобы показать, что алгоритм наискорейшего спуска удовлетворяет условию (3.10), рассмотрим разложение $\mathbf{E}(\mathbf{w}(n+1))$ в ряд Тейлора относительно $\mathbf{w}(n)$ с точностью до членов первого порядка

$$\mathbf{E}(\mathbf{w}(n+1)) = \mathbf{E}(\mathbf{w}(n)) + \mathbf{g}^T(n)\Delta \mathbf{w}(n),$$

которое допустимо при малых значениях параметра η. Подставляя эту формулу в (3.13), получим:

$$\mathbf{E}(\mathbf{w}(n+1)) = \mathbf{E}(\mathbf{w}(n)) - \eta \mathbf{g}^T(n)\mathbf{g}(n) = \mathbf{E}(\mathbf{w}(n)) - \eta ||\mathbf{g}(n)||^2.$$

Из этого выражения видно, что для малых значений параметра скорости обучения η значение функции стоимости уменьшается на каждой итерации. Представленное доказательство истинно только для малых значений η.

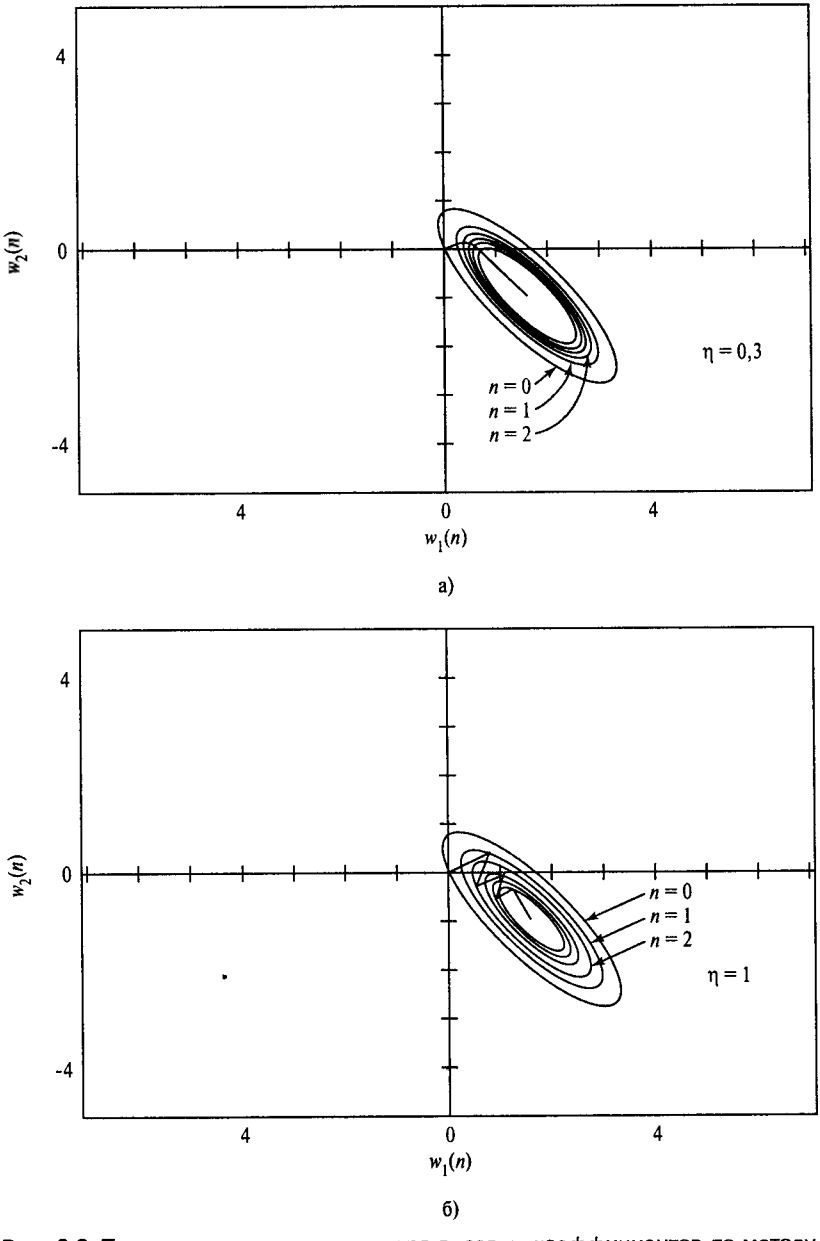


Рис. 3.2. Траектория изменения вектора весовых коэффициентов по методу наискорейшего спуска в двумерном пространстве для двух различных значений параметра скорости обучения $\eta = 0.3$ (а) и $\eta = 1.0$ (б). Координаты w_1 и w_2 являются элементами вектора весов $\mathbf{w}$

Метод наискорейшего спуска сходится к оптимальному значению $\mathbf{w}^*$ достаточно медленно. Кроме того, на скорость сходимости влияет значение параметра η.

- Если параметр η мал, алгоритм *замедляется* (overdamped), и траектория изменения $\mathbf{w}(n)$ соответствует гладкой кривой (рис. 3.2, *а*).
- Если параметр η велик, алгоритм *ускоряется* (underdamped), и траектория $\mathbf{w}(n)$ принимает зигзагообразный вид (рис. 3.2, *б*).
- Если параметр η превосходит некоторое критичное значение, алгоритм становится неустойчивым (т.е. расходящимся).

Метод Ньютона

Основная идея *метода Ньютона* (Newton's method) заключается в минимизации квадратичной аппроксимации функции стоимости $\mathbf{E}(\mathbf{w})$ в точке $\mathbf{w}(n)$. Минимизация выполняется на каждом шаге алгоритма. Используя разложение функции стоимости в ряд Тейлора с точностью до членов второго порядка, можно записать:

$$\begin{aligned}\Delta\mathbf{E}(\mathbf{w}(n)) &= \mathbf{E}(\mathbf{w}(n+1)) - \mathbf{E}(\mathbf{w}(n)) = \\ &= \mathbf{g}^T(n)\Delta\mathbf{w}(n) + 1/2\Delta\mathbf{w}^T(n)\mathbf{H}(n)\Delta\mathbf{w}(n).\end{aligned} \tag{3.14}$$

Здесь, как и ранее, $\mathbf{g}(n)$ — вектор градиента функции $\mathbf{E}(\mathbf{w})$ размерности $m\times 1$, вычисленный в точке $\mathbf{w}(n)$. Матрица $\mathbf{H}(n)$ — *матрица Гессе*, или Гессиан, также вычисленный в точке $\mathbf{w}(n)$ по следующей формуле:

$$\mathbf{H} = \nabla^2\mathbf{E}(\mathbf{w}) = \begin{bmatrix} \frac{\partial^2\mathbf{E}}{\partial w_1^2} & \frac{\partial^2\mathbf{E}}{\partial w_1\partial w_2} & \cdots & \frac{\partial^2\mathbf{E}}{\partial w_1\partial w_m} \\ \frac{\partial^2\mathbf{E}}{\partial w_2\partial w_1} & \frac{\partial^2\mathbf{E}}{\partial w_2^2} & \cdots & \frac{\partial^2\mathbf{E}}{\partial w_2\partial w_m} \\ \cdots & \cdots & \cdots & \\ \frac{\partial^2\mathbf{E}}{\partial w_m\partial w_1} & \frac{\partial^2\mathbf{E}}{\partial w_m\partial w_2} & \cdots & \frac{\partial^2\mathbf{E}}{\partial w_m^2} \end{bmatrix}. \tag{3.15}$$

Для существования Гессиана, определяемого выражением (3.15), функция стоимости должна быть дважды непрерывно дифференцируемой по элементам вектора $\mathbf{w}$. Дифференцируя[2] выражение (3.14) по $\Delta\mathbf{w}$, получим, что инкремент $\Delta\mathbf{E}(\mathbf{w})$ достигает

---

[2] **Дифференцирование по вектору**

Пусть $f(\mathbf{w})$ — действительная функция векторного аргумента $\mathbf{w}$. Производная $f(\mathbf{w})\mathbf{w}$ определяется как вектор

$$\frac{\partial f}{\partial \mathbf{w}} = \left[\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \ldots, \frac{\partial f}{\partial w_m}\right]^T,$$

где m — размериость вектора $\mathbf{w}$. Следующие два случая представляют особый интерес.

Случай 1

Функция $f(\mathbf{w})$ определяется скалярным произведением

$$f(\mathbf{w}) = \mathbf{x}^T\mathbf{w} = \sum_{i=1}^{m} x_i w_i.$$

минимума при условии

$$\mathbf{g}(n) + \mathbf{H}(n)\Delta\mathbf{w}(n) = 0.$$

Разрешая это уравнение относительно $\Delta\mathbf{w}(n)$, получим:

$$\Delta\mathbf{w}(n) = -\mathbf{H}^{-1}(n)\mathbf{g}(n).$$

Таким образом,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta\mathbf{w}(n) = \mathbf{w}(n) - \mathbf{H}^{-1}(n)\boldsymbol{g}(n), \tag{3.16}$$

где $\mathbf{H}^{-1}(n)$ — матрица, обратная Гессиану.

В целом метод Ньютона довольно быстро асимптотически сходится и не приводит к появлению зигзагообразных траекторий, как метод наискорейшего спуска. Однако для обеспечения работоспособности метода Ньютона матрица Гессе $\mathbf{H}(n)$ должна быть *положительно определенной*[3] (positive definite matrix) для всех n. К сожалению, положительную определенность матрицы $\mathbf{H}(n)$ нельзя гарантировать для всех итераций алгоритма. Если Гессиан не является положительно определенной матрицей, метод Ньютона требует некоторой коррекции [124], [854].

В этом случае

$$\frac{\partial f}{\partial w_i} = x_i, i = 1,\ 2,\ \ldots,\ m$$

или, в матричной форме, можно записать:

$$\frac{\partial f}{\partial \mathbf{w}} = \mathbf{x}. \tag{1}$$

Случай 2

Функция f(w) определяется квадратичной формой

$$f(\mathbf{w}) = \mathbf{w}^T\mathbf{R}\mathbf{w} = \sum_{i=1}^{m}\sum_{j=1}^{m} w_i r_{ij} w_j,$$

где r_{ij} — ij-й элемент матрицы **R** размерности $m \times m$. Так как

$$\frac{\partial f}{\partial w_i} = \sum_{j=1}^{m} r_{ij} w_j, \quad i = 1,\ 2,\ \ldots,\ m,$$

то в матричной форме можно записать

$$\frac{\partial f}{\partial \mathbf{w}} = \mathbf{R}\mathbf{w}. \tag{2}$$

Выражения (1) и (2) представляют собой два важных правила дифференцирования вещественных функций по вектору.

[3] **Положительно определенная матрица**

Матрица **R** размерности m×m называется неотрицательно определенной, если она удовлетворяет условию $\mathbf{a}^T\mathbf{R}\mathbf{a} \geq 0$ для любого вектора $\mathbf{a} \in \Re^m$. Если при этом выполняется строгое неравенство, то матрица называется положительно определенной.

Важным свойством положительно определенных матриц является их несингулярность, т.е. существование обратной матрицы $\mathbf{R}^{-1}$. Еще одним важным свойством положительно определенных матриц является то, что все их собственные значения (т.е. корни характеристического уравнения det (**R**)=0) положительны.

Метод Гаусса–Ньютона

Метод Гаусса–Ньютона применяется для минимизации функции стоимости, представленной в виде суммы квадратов ошибок. Пусть

$$\mathbf{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n} e^2(i), \tag{3.17}$$

где коэффициент 1/2 введен для упрощения последующего анализа. Все слагаемые ошибок в этой формуле вычисляются на основании вектора весов $\mathbf{w}$, фиксированного на всем интервале наблюдения $1 \leq i \leq n$.

Сигнал ошибки $e(i)$ является функцией от настраиваемого вектора весов $\mathbf{w}$. Для текущего значения $\mathbf{w}(n)$ зависимость $e(i)$ от $\mathbf{w}$ можно линеаризовать следующим образом:

$$e'(i, \mathbf{w}) = e(i) + \left[\frac{\partial e(i)}{\partial \mathbf{w}}\right]^T_{\mathbf{w}=\mathbf{w}(n)} (\mathbf{w} - \mathbf{w}(n)), \quad i = 1,\ 2,\ \ldots,\ n. \tag{3.18}$$

Эту же формулу можно записать в матричном виде:

$$\mathbf{e}'(n, \mathbf{w}) = \mathbf{e}(n) + \mathbf{J}(n)(\mathbf{w} - \mathbf{w}(n)), \tag{3.19}$$

где $\mathbf{e}(n)$ — вектор ошибки

$$\mathbf{e}(n) = [e(1), e(2), \ldots, e(m)]^T,$$

$\mathbf{J}(n)$ — *матрица Якобиана* ошибки

$$\mathbf{J}(n) = \left[\begin{array}{cccc} \frac{\partial e(1)}{\partial w_1} & \frac{\partial e(1)}{\partial w_2} & \cdots & \frac{\partial e(1)}{\partial w_m} \\ \frac{\partial e(2)}{\partial w_1} & \frac{\partial e(2)}{\partial w_2} & \cdots & \frac{\partial e(2)}{\partial w_m} \\ \cdots & \cdots & \cdots\cdots & \\ \frac{\partial e(n)}{\partial w_1} & \frac{\partial e(n)}{\partial w_2} & \cdots & \frac{\partial e(n)}{\partial w_m} \end{array}\right]_{\mathbf{w}=\mathbf{w}(n)}. \tag{3.20}$$

Якобиан — это транспонированная матрица градиента $\nabla \mathbf{e}(n)$

$$\nabla E(n) = [\nabla e(1), \nabla e(2), \ldots, \nabla e(n)].$$

Обновленный вектор $\mathbf{w}(n+1)$ можно записать в следующем виде:

$$\mathbf{w}(n+1) = \arg\min_{\mathbf{w}} \left\{\frac{1}{2}\left\|\mathbf{e}'(n, \mathbf{w})\right\|^2\right\}. \tag{3.21}$$

Используя формулу (3.19) для оценки квадратичной Евклидовой нормы $\|\mathbf{e}'(n, \mathbf{w})\|^2$, получим:

$$\frac{1}{2}\|\mathbf{e}'(n, \mathbf{w})\|^2 = \frac{1}{2}\|\mathbf{e}(n)\|^2 + \mathbf{e}^T(n)\mathbf{J}(n)(\mathbf{w} - \mathbf{w}(n)) + \\ + \frac{1}{2}(\mathbf{w} - \mathbf{w}(n))^T\mathbf{J}^T(n)\mathbf{J}(n)(\mathbf{w} - \mathbf{w}(n)).$$

Дифференцируя это выражение по $\mathbf{w}$ и приравнивая результат к нулю, получим:

$$\mathbf{J}^T(n)\mathbf{e}(n) + \mathbf{J}^T(n)\mathbf{J}(n)(\mathbf{w} - \mathbf{w}(n)) = \mathbf{0}.$$

Разрешая это уравнение относительно $\mathbf{w}$ и учитывая (3.21), можно записать:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - (\mathbf{J}^T(n)\mathbf{J}(n))^{-1}\mathbf{J}^T(n)\mathbf{e}(n). \tag{3.22}$$

Эта формула описывает метод Гаусса–Ньютона в чистом виде.

В отличие от метода Ньютона, требующего знания матрицы Гессиана для функции стоимости $\mathbf{E}(n)$, метод Гаусса–Ньютона требует только знания матрицы Якобиана вектора ошибки $\mathbf{e}(n)$. Тем не менее для реализации итеративного метода Гаусса–Ньютона матрица произведения $\mathbf{J}^T(n)\mathbf{J}(n)$ должна быть несингулярной.

Возвращаясь к предыдущей формуле, можно заметить, что матрица $\mathbf{J}^T(n)\mathbf{J}(n)$ всегда является неотрицательно определенной. Для обеспечения несингулярности Якобиан $\mathbf{J}(n)$ должен иметь ранг n (т.е. n строк матрицы $\mathbf{J}(n)$ в формуле (3.20) должны быть линейно-независимы). К сожалению, это условие выполняется не всегда. Для обеспечения необходимого ранга матрицы $\mathbf{J}(n)$, общей к произведению $\mathbf{J}^T(n)\mathbf{J}(n)$, зачастую добавляют диагональную матрицу $\delta\mathbf{I}$, где $\mathbf{I}$ — единичная матрица. Параметр δ является малой положительной константой, обеспечивающей положительную определенность матрицы $\mathbf{J}^T(n)\mathbf{J}(n) + \delta\mathbf{I}$ для всех n.

Исходя из вышесказанного, уравнение метода Гаусса–Ньютона можно записать в несколько видоизмененном виде:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - (\mathbf{J}^T(n)\mathbf{J}(n) + \delta\mathbf{I})^{-1}\mathbf{J}^T(n)\mathbf{e}(n). \tag{3.23}$$

Влияние этой модификации постепенно ослабляется с увеличением количества итераций n. Обратите внимание, что рекурсивное соотношение (3.23) является решением задачи минимизации *модифицированной* функции стоимости:

$$\mathbf{E}(\mathbf{w}) = \frac{1}{2}\left\{\delta\|\mathbf{w} - \mathbf{w}(n)\|^2 + \sum_{i=1}^{n} e^2(i)\right\}, \tag{3.24}$$

где $\mathbf{w}(n)$ — *текущее значение* (current value) вектора весовых коэффициентов $\mathbf{w}(i)$.

Ознакомившись с основными методами оптимизации, можно вплотную заняться вопросами линейной адаптивной фильтрации.

3.4. Линейный фильтр, построенный по методу наименьших квадратов

Как и следует из самого названия, *линейный фильтр, построенный по методу наименьших квадратов* (linear least-squares filter), имеет две отличительные особенности. Во-первых, он строится для отдельного линейного нейрона (см. рис. 3.1, *б*). Во-вторых, функция стоимости $\mathbf{E}(\mathbf{w})$, используемая для создания этого фильтра, представляет собой сумму квадратов ошибок и определяется в соответствии с формулой (3.17). Принимая во внимание формулы (3.3) и (3.4), для вектора ошибки $\mathbf{e}(n)$ можно записать следующее соотношение:

$$\mathbf{e}(n) = \mathbf{d}(n) - [\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(n)]^T \mathbf{w}(n) = \mathbf{d}(n) - \mathbf{X}(n)\mathbf{w}(n), \tag{3.25}$$

где $\mathbf{d}(n)$ — вектор желаемого *отклика* размерности n

$$\mathbf{d}(n) = [d(1), d(2), \ldots, d(n)]^T,$$

$\mathbf{X}(n)$ — матрица данных размерности $n \times \quad m$

$$\mathbf{X}(n) = [\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(n)]^T.$$

Дифференцируя выражение (3.25) по $\mathbf{w}(n)$, получим матрицу градиента

$$\nabla E(n) = -\mathbf{X}^T(n).$$

Следовательно, Якобиан $\mathbf{e}(n)$ можно записать в следующем виде:

$$\mathbf{J}(n) = -\mathbf{X}(n). \tag{3.26}$$

Так как уравнение ошибки (3.19) является линейным относительно вектора весовых коэффициентов $\mathbf{w}(n)$, метод Гаусса–Ньютона сходится за одну итерацию. Подставляя (3.25) и (3.26) в (3.22), получим:

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + (\mathbf{X}^T(n)\mathbf{X}(n))^{-1}\mathbf{X}^T(n)(\mathbf{d}(n) - \mathbf{X}(n)\mathbf{w}(n)) = \\ &= (\mathbf{X}^T(n)\mathbf{X}(n))^{-1}\mathbf{X}^T(n)\mathbf{d}(n).\end{aligned} \tag{3.27}$$

Выражение $(\mathbf{X}^T(n)\mathbf{X}(n))^{-1}\mathbf{X}^T(n)$ называют *псевдообратной* матрицей для матрицы данных $\mathbf{X}(n)$ и обозначают следующим образом [368], [434]:

$$\mathbf{X}^+(n) = (\mathbf{X}^T(n)\mathbf{X}(n))^{-1}\mathbf{X}^T(n). \tag{3.28}$$

Исходя из этого, выражение (3.27) можно переписать в более компактном виде:

$$\mathbf{w}(n+1) = \mathbf{X}^+(n)\mathbf{d}(n). \tag{3.29}$$

Это выражение вербально можно описать следующим образом: "Вектор весовых коэффициентов $\mathbf{w}(n+1)$ является решением линейной задачи фильтрации, решаемой по методу наименьших квадратов на интервале наблюдения длительности n".

Фильтр Винера как ограниченная форма линейного фильтра, построенного по методу наименьших квадратов, для эргодической среды

Частным случаем, представляющим особый интерес, является получение вектора входного сигнала $\mathbf{x}(i)$ и желаемого отклика $d(i)$ из *эргодической стационарной среды* (ergodic). Для этого случая вместо усреднения по времени можно использовать математическое ожидание или усреднение по множеству [380]. Такая среда частично описывается следующими статистическими характеристиками второго порядка.

- *Матрица корреляции* (correlation matrix) $\mathbf{R}_\mathbf{x}$ вектора входного сигнала $\mathbf{x}(i)$.
- *Вектор взаимной корреляции* (cross-correlation vector) $\mathbf{r}_{\mathbf{x}d}$ между вектором входного сигнала $\mathbf{x}(i)$ и ожидаемого отклика $d(i)$.

Эти величины определяются следующими выражениями:

$$\mathbf{R}_\mathbf{x} = E\left[\mathbf{x}(i)\mathbf{x}^T(i)\right] = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}(n)\mathbf{x}^T(i) = \lim_{n\to\infty}\frac{1}{n}\mathbf{X}^T(n)\mathbf{X}(n), \tag{3.30}$$

$$\mathbf{r}_{\mathbf{x}d} = E\left[\mathbf{x}(i)d(i)\right] = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}(i)d(i) = \lim_{n\to\infty}\frac{1}{n}\mathbf{X}^T(n)\mathbf{d}(n), \tag{3.31}$$

где E — статистический оператор математического ожидания. Тогда решение линейной задачи фильтрации по методу наименьших квадратов (3.27) можно переписать в следующем виде:

$$\begin{aligned}\mathbf{w}_o &= \lim_{n\to\infty}\mathbf{w}(n+1) = \lim_{n\to\infty}(\mathbf{X}^T(n)\mathbf{X}(n))^{-1}\mathbf{X}^T(n)\mathbf{d}(n) = \\ &= \lim_{n\to\infty}\frac{1}{n}(\mathbf{X}^T(n)\mathbf{X}(n))^{-1}\lim_{n\to\infty}\frac{1}{n}\mathbf{X}^T(n)\mathbf{d}(n) = \mathbf{R}_\mathbf{x}^{-1}\mathbf{r}_{\mathbf{x}d},\end{aligned} \tag{3.32}$$

где $\mathbf{R}_x^{-1}$ — обратная матрица для матрицы корреляции $\mathbf{R}_x$. Вектор весовых коэффициентов $\mathbf{w}_o$ называют *Винеровским решением* линейной задачи оптимальной фильтрации в честь Норберта Винера (Norbert Wiener), который внес весомый вклад в ее решение [434], [1144]. Исходя из этого, можно сформулировать следующее утверждение.

Для эргодического процесса линейный фильтр, построенный по методу наименьших квадратов, асимптотически сходится к фильтру Винера по мере приближения количества наблюдений к бесконечности.

Для построения фильтра Винера необходимо знать статистические характеристики второго порядка: матрицу корреляции $\mathbf{R}_x$ для вектора входного сигнала $\mathbf{x}(n)$ и вектора взаимной корреляции $\mathbf{r}_{xd}$ между $\mathbf{x}(n)$ и желаемым откликом $d(n)$. Однако эта информация на практике зачастую недоступна. В неизвестной среде можно использовать *линейный адаптивный фильтр* (linear adaptive filter). Под термином "адаптивность" понимается возможность настраивать свободные параметры фильтра в соответствии со статистическими вариациями среды. Наиболее популярным алгоритмом такой настройки на непрерывной основе является алгоритм минимизации среднеквадратической ошибки, который самым тесным образом связан с фильтром Винера.

3.5. Алгоритм минимизации среднеквадратической ошибки

Алгоритм минимизации среднеквадратической ошибки (least-mean-square — LMS) основан на использовании дискретных значений функции стоимости:

$$\mathbf{E}(\mathbf{w}) = 1/2e^2(n), \tag{3.33}$$

где $e(n)$ — сигнал ошибки, измеренный в момент времени n. Дифференцируя $\mathbf{E}(\mathbf{w})$ по вектору весов $\mathbf{w}$, получим:

$$\frac{\partial\, \mathbf{E}(\mathbf{w})}{\partial\, \mathbf{w}} = e(n)\frac{\partial\, e(n)}{\partial\, \mathbf{w}}. \tag{3.34}$$

Как и в случае линейного фильтра, построенного по методу наименьших квадратов, алгоритм минимизации среднеквадратической ошибки работает с линейным нейроном. Исходя из этого, сигнал ошибки можно записать в следующем виде:

$$\mathbf{e}(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n). \tag{3.35}$$

Следовательно,

$$\frac{\partial\, e(n)}{\partial\, \mathbf{w}(n)} = -\mathbf{x}(n),$$
$$\frac{\partial\, \mathbf{E}(\mathbf{w})}{\partial\, \mathbf{w}(n)} = -\mathbf{x}(n)e(n).$$

Используя полученный результат, можно *оценить* вектор градиента

$$\hat{\mathbf{g}}(n) = -\mathbf{x}(n)e(n). \tag{3.36}$$

И, наконец, используя формулу (3.36) для вектора градиента в методе наискорейшего спуска (3.12), можно сформулировать алгоритм минимизации среднеквадратической ошибки в следующем виде:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta\, \mathbf{x}(n)e(n), \tag{3.37}$$

где η — параметр скорости обучения. Контур обратной связи для вектора весов $\hat{\mathbf{w}}(n)$ в алгоритме минимизации среднеквадратической ошибки ведет себя как *низкочастотный фильтр* (low-pass filter), передающий низкочастотные компоненты сигнала ошибки и задерживающий его высокочастотные составляющие [434]. Усредненная временная константа этой фильтрации обратно пропорциональна параметру скорости обучения η. Следовательно, при малых значениях η процесс адаптации будет продвигаться медленно. При этом алгоритм минимизации среднеквадратической ошибки будет запоминать большее количество предшествующих данных, а значит, более точной будет фильтрация. Другими словами, величина, обратная параметру скорости обучения η, является *мерой памяти* (measure of memory) алгоритма минимизации среднеквадратической ошибки.

В формуле (3.37) вместо $\mathbf{w}(n)$ мы использовали $\hat{\mathbf{w}}(n)$. Этим подчеркивается тот факт, что алгоритм минимизации среднеквадратической ошибки только *оценивает* (estimate) вектор весовых коэффициентов на основе метода наискорейшего спуска. Следовательно, используя алгоритм минимизации среднеквадратической ошибки, можно пожертвовать одной из отличительных особенностей алгоритма наискорейшего спуска. В последнем вектор весовых коэффициентов $\mathbf{w}(n)$ изменяется по детерминированной траектории в пространстве весов при заранее выбранном параметре η. В противоположность этому в алгоритме минимизации среднеквадратической ошибки вектор $\hat{\mathbf{w}}(n)$ перемещается по случайной траектории. По этой причине алгоритм минимизации среднеквадратической ошибки иногда называют стохастическим градиентным алгоритмом. При стремлении количества итераций в алгоритме LMS к бесконечности вектор $\hat{\mathbf{w}}(n)$ выписывает хаотичную траекторию (в соответствии с

ТАБЛИЦА 3.1. Краткое описание алгоритма минимизации среднеквадратической ошибки

Обучающий пример	Вектор входного сигнала = $\mathbf{x}(n)$ Желаемый отклик = $d(n)$
Выбираемый пользователем параметр	η
Инициализация весов	$\hat{\mathbf{w}}(0) = 0$
Вычислительная схема	Для $n = 1, 2, \ldots$ $e(n) = d(n) - \mathbf{x}^T(n)\hat{\mathbf{w}}(n)$, $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta\, \mathbf{x}(n)e(n)$

принципами броуновского движения) вокруг Винеровского решения $\mathbf{w}_o$. Важно отметить, что, в отличие от метода наискорейшего спуска, алгоритм минимизации среднеквадратической ошибки *не требует* знания статистических характеристик окружающей среды.

Краткое описание алгоритма минимизации среднеквадратической ошибки представлено в табл. 3.1, из которой ясно видна его простота. Из таблицы видно, что для *инициализации* (initialization) алгоритма достаточно обнулить его начальный вектор весовых коэффициентов.

Граф передачи сигнала для алгоритма минимизации среднеквадратической ошибки

Объединяя формулы (3.35) и (3.37), эволюцию вектора весов в алгоритме минимизации среднеквадратической ошибки можно представить в следующем виде:

$$\begin{aligned}\hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n) + \eta\, \mathbf{x}(n)[d(n) - \mathbf{x}^T(n)\hat{\mathbf{w}}(n)] = \\ &= [\mathbf{I} - \eta\, \mathbf{x}(n)\mathbf{x}^T(n)]\hat{\mathbf{w}}(n) + \eta\, \mathbf{x}(n)d(n),\end{aligned} \tag{3.38}$$

где $\mathbf{I}$ — единичная матрица. При использовании алгоритма минимизации среднеквадратической ошибки

$$\hat{\mathbf{w}}(n) = z^{-1}[\hat{\mathbf{w}}(n+1)], \tag{3.39}$$

где z^{-1} — *оператор единичной задержки*, реализующей память алгоритма. Используя выражения (3.38) и (3.39), алгоритм минимизации среднеквадратической ошибки можно представить в виде графа передачи сигнала, показанного на рис. 3.3. На этом графе видно, что алгоритм минимизации среднеквадратической ошибки является всего лишь частным случаем *стохастической системы с обратной связью* (stochastic feedback system). Наличие обратной связи оказывает первостепенное влияние на сходимость алгоритма LMS.

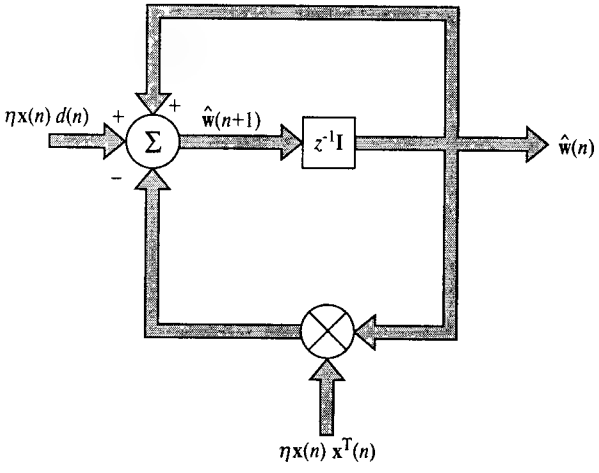


Рис. 3.3. Граф передачи сигнала для алгоритма минимизации среднеквадратической ошибки

Условия сходимости алгоритма LMS

Из теории управления известно, что устойчивость системы с обратной связью определяется параметрами обратной связи. На рис. 3.3 видно, что нижний контур обратной связи обеспечивает изменение поведения алгоритма минимизации среднеквадратической ошибки. В частности, пропускная способность контура обратной связи определяется двумя параметрами: коэффициентом скорости обучения η и вектором входного сигнала $\mathbf{x}(n)$. Таким образом, можно заключить, что сходимость (и устойчивость) алгоритма минимизации среднеквадратической ошибки зависит от статистических характеристик входного вектора $\mathbf{x}(n)$ и значения параметра η. Другими словами, для каждой среды, из которой поступают входные векторы, следует подбирать собственный параметр скорости обучения η, который обеспечит сходимость алгоритма LMS.

Первым критерием сходимости алгоритма минимизации среднеквадратической ошибки является *сходимость в среднем* (convergence of the mean):

$$E[\hat{\mathbf{w}}(n)] \to \mathbf{w}_o \text{ при } n \to \infty, \tag{3.40}$$

где $\mathbf{w}_o$ — решение Винера. К сожалению, такой критерий сходимости не имеет большого практического значения, так как последовательность произвольных векторов, имеющих среднее значение 0, также будет удовлетворять этому условию.

С практической точки зрения вопрос устойчивости играет роль именно в смысле *среднеквадратической сходимости* (convergence of the mean square):

$$E[e^2(n)] \to \text{const при } n \to \infty. \tag{3.41}$$

К сожалению, детальный анализ сходимости в смысле среднеквадратического значения для алгоритма LMS чрезвычайно сложен. Чтобы его математически упростить, сделаем следующие предположения.

1. Векторы входных сигналов $\mathbf{x}(1)$, $\mathbf{x}(2)$, ... являются статистически независимыми друг от друга.
2. В момент времени n вектор входного сигнала $\mathbf{x}(n)$ является статистически независимым от всех предыдущих желаемых откликов, т.е. $d(1)$, $d(2)$, ..., $d(n-1)$.
3. В момент времени n желаемый отклик $d(n)$ зависит от вектора $\mathbf{x}(n)$, но статистически не зависит от всех предыдущих значений желаемого отклика.
4. Вектор входного сигнала $\mathbf{x}(n)$ и желаемый отклик $d(n)$ выбираются из множества, подчиняющегося распределению Гаусса.

Статистический анализ алгоритма минимизации среднеквадратической ошибки при наличии вышеуказанных допущений получил название *теории независимости* (independence theory) [1138].

С учетом допущений теории независимости и достаточно малого значения параметра скорости обучения в [434] показано, что алгоритм минимизации среднеквадратической ошибки сходится в смысле среднеквадратического значения, если η удовлетворяет условию

$$0 < \eta < \frac{2}{\lambda_{\max}}, \tag{3.42}$$

где λ_{max} — *наибольшее собственное значение* (largest eigenvalue) матрицы корреляции $\mathbf{R}_{\mathbf{x}}$. В типичных приложениях алгоритма LMS значение $\lambda_{\max}$, как правило, не известно. Чтобы обойти эту сложность, в качестве оценки сверху значения $\lambda_{\max}$ можно использовать *след* (trace) матрицы $\mathbf{R}_{\mathbf{x}}$. В этом случае условие (3.42) можно переписать в следующем виде:

$$0 < \eta < \frac{2}{\mathrm{tr}[\mathbf{R}_{\mathbf{x}}]}, \tag{3.43}$$

где $\mathrm{tr}[\mathbf{R}_{\mathbf{x}}]$ — след матрицы $\mathbf{R}_{\mathbf{x}}$. По определению след квадратной матрицы определяется как сумма ее диагональных элементов. Так как каждый из диагональных элементов матрицы корреляции $\mathbf{R}_{\mathbf{x}}$ является среднеквадратическим значением соответствующего входного сигнала, условие сходимости алгоритма минимизации среднеквадратической ошибки можно сформулировать в виде

$$0 < \eta < \frac{2}{\text{сумма среднеквадратических значений входных сигналов}}. \tag{3.44}$$

Если параметр скорости обучения удовлетворяет этому условию, то алгоритм минимизации среднеквадратической ошибки сходится также и в смысле среднего значения. Это значит, что сходимость в смысле среднеквадратического значения является достаточным условием сходимости в среднем. Обратное утверждение верно не всегда.

Преимущества и недостатки алгоритма LMS

Важным преимуществом алгоритма минимизации среднеквадратической ошибки является его простота (см. табл. 3.1). Кроме того, этот алгоритм независим от модели и, таким образом, является робастным. Это значит, что при малой неопределенности в модели и небольших возмущениях (с малой энергией) сигнал ошибки также будет невелик. В более строгих математических терминах алгоритм минимизации среднеквадратической ошибки является оптимальным согласно *минимаксному критерию* (H^{∞}) (minimax criterion) [426], [427]. Основная идея оптимальности в смысле H^{∞} — это обеспечить наилучшее выполнение пессимистического сценария[4]. Ее можно сформулировать так.

Если вы не знаете, с чем столкнулись, предположите наихудшее и оптимизируйте решение.

Долгое время алгоритм минимизации среднеквадратической ошибки рассматривался как частный случай алгоритма градиентного спуска. Однако оптимальность алгоритма минимизации среднеквадратической ошибки в смысле H^{∞} придает ему более устойчивое положение в рассматриваемой предметной области. В частности, он обеспечивает приемлемые результаты не только в стационарной, но и в нестационарной среде. Под "нестационарностью" среды подразумевается то, что ее статистические характеристики изменяются во времени. В такой среде оптимальное решение Винера зависит от времени, а алгоритм минимизации среднеквадратической ошибки выполняет дополнительную задачу *отслеживания* (tracking) изменения параметров фильтра Винера.

Основным ограничением алгоритма минимизации среднеквадратической ошибки является низкая скорость сходимости и чувствительность к изменению собственных значений матрицы входных сигналов [434]. Для сходимости алгоритма минимизации среднеквадратической ошибки обычно требуется в 10 раз больше итераций, чем размерность пространства входных сигналов. Медленная сходимость становится действительно серьезной преградой, если размерность пространства входных данных очень велика. Что же касается чувствительности, то наибольшая чувствительность алгоритма проявляется к изменению *числа обусловленности* (condition number) или разброса собственных чисел матрицы корреляции $\mathbf{R}_{\mathbf{x}}$ входного вектора $\mathbf{x}$. Число обусловленности $\chi(\mathbf{R}_{\mathbf{x}})$ определяется следующим выражением:

---

[4] ***Робастность*** Критерий H^* введен Зеймсом (Zames) в [1179] и развит в [1180]. Этот критерий обсуждается и в некоторых других работах, в частности в [265], [381], [425].

$$\chi(\mathbf{R}_{\mathbf{x}}) = \frac{\lambda_{\max}}{\lambda_{\min}}, \tag{3.45}$$

где $\lambda_{\min}$ и $\lambda_{\max}$ — соответственно минимальное и максимальное собственные числа матрицы $\mathbf{R}_{\mathbf{x}}$. Чувствительность алгоритма минимизации среднеквадратической ошибки к вариациям числа обусловленности $\chi(\boldsymbol{R}_{\mathbf{x}})$ становится действительно опасной, когда обучающая выборка, которой принадлежит вектор $\mathbf{x}(n)$, является плохо *обусловленной*, т.е. число обусловленности $\chi(\mathbf{R}_{\mathbf{x}})$ достаточно велико[5].

Заметим, что в алгоритме минимизации среднеквадратической ошибки *матрица Гессиана*, определяемая как вторая производная от функции стоимости $\mathbf{E}(\mathbf{w})$ по вектору $\mathbf{w}$, эквивалентна матрице корреляции $\mathbf{R}_{\mathbf{x}}$ (см. задачу 3.8). Таким образом, в дальнейших рассуждениях мы будем употреблять оба этих термина в одинаковом значении.

3.6. Графики процесса обучения

Одним из самых информативных способов проверки сходимости алгоритма минимизации среднеквадратической ошибки и всех адаптивных фильтров в целом является построение графиков процесса обучения или так называемых *кривых обучения* (learning curve) для различных условий внешней среды. Кривая обучения является *графиком изменения среднеквадратического значения ошибки оценивания* $\mathbf{E}_{\text{av}}(n)$ в зависимости от *количества итераций* n.

Представим себе эксперимент, проводимый над множеством адаптивных фильтров, когда каждый из них работает под управлением отдельного алгоритма. Предполагается, что начальные условия и принципы работы всех алгоритмов одинаковы. Различие между ними определяется *случайностью* выбора вектора входного сигнала $\mathbf{x}(n)$ и желаемого отклика $d(n)$ из имеющейся обучающей выборки. Для каждого из фильтров строится график изменения среднеквадратической ошибки оценивания (т.е. разности между желаемым откликом и фактическим выходным сигналом фильтра) относительно количества итераций. Полученное таким образом множество кривых обучения состоит из *зашумленных* (noisy) графиков экспоненциального типа. Наличие шума связано со стохастической природой адаптивного фильтра. Чтобы построить *усредненную по этому множеству кривую обучения* (ensemble averaged learning curve) (т.е. график $\mathbf{E}_{\text{av}}(n)$ относительно n), нужно вычислить средние значения по всем кривым, участвующим в эксперименте. Это способствует снижению влияния шума на результат.

---

[5] Чтобы преодолеть недостатки алгоритма LMS, т.е. повысить скорость сходимости и снизить чувствительность к вариациям числа обусловленности матрицы $\mathbf{R}_{\mathbf{x}}$, можно использовать *рекурсивный алгоритм наименьших квадратов (RLS)*, который является следствием линейного фильтра, построенного по методу наименьших квадратов и описаного в разделе 3.4. Алгоритм RLS является частным случаем фильтра Калмана — известного линейного оптимального фильтра для нестационарной среды. Следует отметить, что фильтр Калмана использует все предыдущие данные вплоть до времени начала вычислений. Более подробно алгоритм RLS и его связи с фильтром Калмана описываются в [434]. Кроме того, фильтр Калмана рассматривается в главе 15.

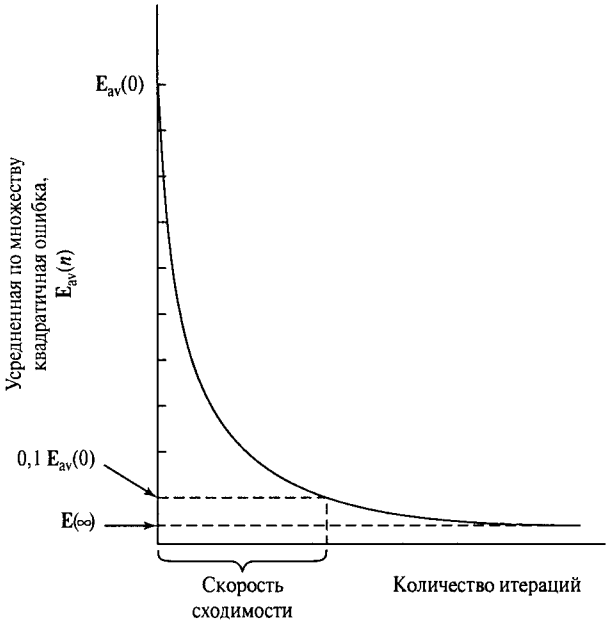


Рис. 3.4. Идеализированная кривая обучения алгоритма LMS

В предположении устойчивости адаптивного фильтра усредненная по множеству кривая обучения начинается с достаточно большого значения $\mathbf{E}_{av}(0)$, определяемого начальными условиями, и затем ее значения уменьшаются с некоторой скоростью, зависящей от типа фильтра, а в пределе сходятся к некоторому устойчивому значению $\mathbf{E}_{av}(\infty)$ (рис. 3.4). Основываясь на этой кривой, можно определить *скорость сходимости* (rate of convergence) адаптивного фильтра, определяемую как число итераций, необходимых для того, чтобы для любого произвольного начального значения $\mathbf{E}_{av}(0)$ величина $\mathbf{E}_{av}(n)$ уменьшилась в 10 раз.

Еще одной полезной характеристикой адаптивного фильтра, получаемой из усредненной кривой обучения, является *рассогласование* (misadjustment), обозначаемое символом **M**. Пусть $\mathbf{E}_{min}$ — минимальная среднеквадратическая ошибка, обеспечиваемая фильтром Винера, построенным на основе известных значений матрицы корреляции $\mathbf{R}_x$ и вектора взаимной корреляции $\mathbf{r}_{xd}$. Исходя из этого, рассогласование можно определить следующим образом [434], [1144]:

$$\mathbf{M} = \frac{\mathbf{E}(\infty) - \mathbf{E}_{min}}{\mathbf{E}_{min}} = \frac{\mathbf{E}(\infty)}{\mathbf{E}_{min}} - 1. \tag{3.46}$$

Рассогласование **M** является безразмерной величиной — просто мерой измерения близости адаптивного фильтра к оптимальному в смысле среднеквадратической ошибки. Чем ближе значение **M** к единице, тем более *точной* является адаптивная фильтрация алгоритма. Одним из вариантов является представление рассогла-

сования в процентах. Например, рассогласование в 10% означает, что адаптивный фильтр работает со среднеквадратической ошибкой (после завершения адаптации), на 10% превышающей минимальную среднеквадратическую ошибку, обеспечиваемую соответствующим фильтром Винера. На практике такая точность обычно считается удовлетворительной.

Еще одной важной характеристикой алгоритма минимизации среднеквадратической ошибки является *время установки* (setting time). К сожалению, однозначного определения этой величины не существует. Например, кривую обучения можно грубо аппроксимировать экспоненциальной функцией с *усредненной временной константой* (average time constant) τ_{av}. Чем меньше значение τ_{av}, тем короче время установки (т.е. тем быстрее алгоритм минимизации среднеквадратической ошибки будет сходиться к установившемуся состоянию).

В качестве неплохой аппроксимации величины рассогласования **M** можно порекомендовать параметр скорости обучения η, который прямо пропорционален ей. Что же касается величины τ_{av}, то она обратно пропорциональна параметру скорости обучения [434], [1144]. Таким образом, получается противоречие: если уменьшить параметр скорости обучения для уменьшения рассогласования, то увеличивается время установки алгоритма LMS. Следовательно, для ускорения процесса обучения нужно увеличивать коэффициент скорости обучения, однако следует учесть, что одновременно с этим будет увеличиваться и рассогласование. Выбору параметра η нужно уделять большое внимание, так как он является основной величиной в алгоритме LMS, отвечающей за его общую производительность.

3.7. Изменение параметра скорости обучения по модели отжига

Сложность работы с алгоритмом минимизации среднеквадратической ошибки связана с тем фактом, что параметр скорости обучения является эмпирической константой. Ее можно не изменять в течение всего процесса обучения, т.е.

$$\eta(n) = \eta_0 \text{ для всех } n. \tag{3.47}$$

Это простейший способ задания коэффициента скорости обучения. В отличие от него в методах *стохастической аппроксимации* (stochastic approximation), берущих свое начало в классической статье [892], параметр интенсивности обучения изменяется со временем. Одной из самых распространенных форм изменения этого параметра, описанных в литературе по стохастической аппроксимации, является следующая:

$$\eta\,(n) = \frac{c}{n}, \tag{3.48}$$

где c — константа. Такого выбора достаточно, чтобы гарантировать сходимость алгоритма стохастической аппроксимации [607], [665]. Если константа c велика, существует опасность выхода алгоритма из-под контроля на первых шагах аппроксимации (при маленьких n). В качестве альтернативы выбору (3.47) и (3.48) можно использовать так называемый подход на основе поиска и сходимости, впервые предложенный в [238]:

$$\eta(n) = \frac{\eta_0}{1 + (n/\tau)}, \tag{3.49}$$

где η_0 и τ — заданные пользователем константы. На первых шагах адаптации число n является малым по сравнению с *константой времени поиска* (search time constant) τ. Поэтому параметр скорости $\eta(n)$ практически равен константе η_0, и алгоритм ведет себя как стандартный алгоритм минимизации среднеквадратической ошибки (рис. 3.5). Выбирая большое значение η_0(естественно, в допустимых пределах), можно надеяться, что настраиваемые весовые коэффициенты фильтра будут находиться вблизи "хороших" значений. Таким образом, при количестве итераций n, значительно превосходящих константу времени поиска τ, параметр интенсивности скорости $\eta(n)$ будет сходиться к функции c/n, где $c = \tau\eta_0$ (рис. 3.5). Алгоритм при этом будет вести себя как обычный алгоритм стохастической аппроксимации, а веса будут сходиться к своим оптимальным значениям. Таким образом, изменение коэффициента скорости обучения на основе метода поиска и сходимости обеспечивает сочетание полезных свойств обычного алгоритма минимизации среднеквадратической ошибки с методами традиционной теории стохастической аппроксимации.

3.8. Персептрон

Итак, мы переходим ко второй части настоящей главы, в которой речь пойдет о персептроне Розенблатта (далее — просто *персептрон* (perceptron)). Если алгоритм LMS, описанный в предыдущем разделе, разработан для линейного нейрона, то персептрон строится для нелинейного, а именно *модели нейрона Мак-Каллока–Питца*. Из главы 1 известно, что такая нейронная модель состоит из линейного сумматора и ограничителя (реализованного в виде пороговой функции вычисления знака) (рис. 3.6). Суммирующий узел этой нейронной модели вычисляет линейную комбинацию входных сигналов, поступающих на синапсы с учетом внешнего возмущения (порога). Полученная сумма (так называемое *индуцированное локальное поле* (induced local field)) передается на узел ограничителя. Таким образом, выход нейрона принимает значение $+1$, если сигнал на выходе сумматора положителен, и -1, если отрицателен.

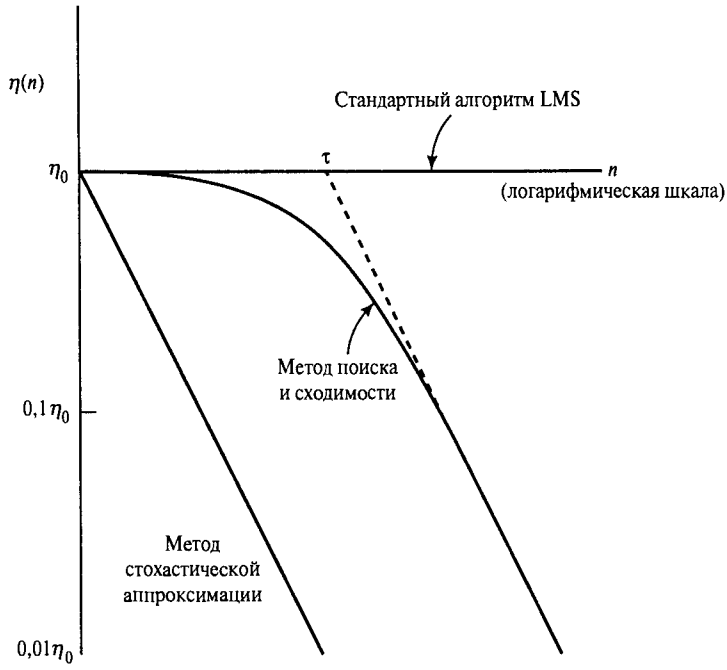


Рис. 3.5. Изменение коэффициента скорости обучения по методу моделирования отжига

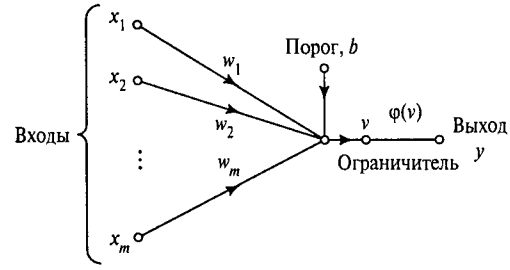


Рис. 3.6. Граф передачи сигнала для персептрона

На диаграмме передачи сигнала (рис. 3.6) синаптические веса персептрона обозначены $w_1, w_2, \ldots, w_m$, сигналы, поступающие на вход персептрона, обозначены $x_1, x_2, \ldots, x_m$, а пороговое значение — b. Исходя из структуры модели, можно заключить, что входной сигнал ограничителя (т.е. индуцированное локальное поле) нейрона определяется выражением

$$v = \sum_{i=1}^{m} w_i x_i + b. \tag{3.50}$$

Целью персептрона является корректное отнесение множества внешних стимулов $x_1, x_2, \ldots, x_m$ к одному из двух классов: $\mathbf{C}_1$ или $\mathbf{C}_2$. Решающее правило такой классификации заключается в следующем: входной сигнал относится к классу $\mathbf{C}_1$, если выход y равен $+1$, и к классу $\mathbf{C}_2$ в противном случае (если выход равен -1).

Чтобы глубже изучить поведение классификатора, целесообразно построить карту областей решения в m-мерном пространстве сигналов, определяемом переменными $x_1, x_2, \ldots, x_m$. В простейшем случае (когда в качестве классификатора выступает персептрон) имеются всего две области решения, разделенные гиперплоскостью, определяемой формулой

$$\sum_{i=1}^{m} w_i x_i + b = 0. \tag{3.51}$$

Это проиллюстрировано на рис. 3.7 для случая двух переменных, x_1 и x_2, когда разделяющая гиперплоскость вырождается в прямую. Точки (x_1, x_2), лежащие выше этой прямой, относятся к классу $\mathbf{C}_1$, а точки, расположенные ниже прямой, принадлежат классу $\mathbf{C}_2$. Обратите внимание, что пороговое значение определяет смещение разделяющей поверхности по отношению к началу координат.

Синаптические веса персептрона $w_1, w_2, \ldots, w_m$ можно адаптировать итеративным методом. В частности, для настройки весовых коэффициентов можно использовать алгоритм, основанный на коррекции ошибок и получивший название *алгоритма сходимости персептрона* (perceptron convergence algorithm).

3.9. Теорема о сходимости персептрона

Чтобы вывести алгоритм обучения персептрона, основанный на коррекции ошибок, удобно построить модифицированный граф передачи сигнала (рис. 3.8). В этой модели, которая эквивалентна модели нейрона, показанной на рис. 3.6, порог $b(n)$ рассматривается как синаптический вес связи с фиксированным входным сигналом $+1$. Это можно описать следующим входным вектором размерности $(m+1)$:

$$\mathbf{x}(n) = [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T,$$

где n — номер итерации алгоритма. Аналогично можно определить $(m+1)$-мерный вектор весовых коэффициентов:

$$\mathbf{w}(n) = [b(n), w_1(n), w_2(n), \ldots, w_m(n)]^T.$$

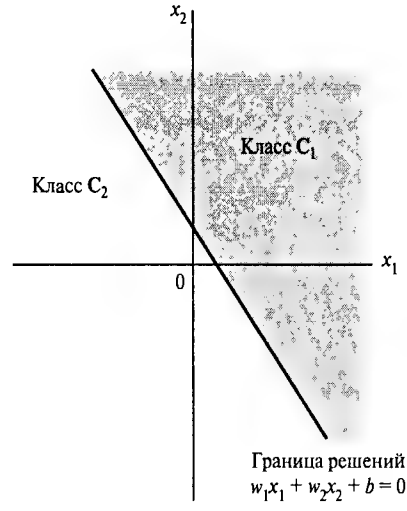


Рис. 3.7. Разделяющая поверхность в виде гиперплоскости (в данном случае — прямой) для двумерной задачи классификации образов на два класса

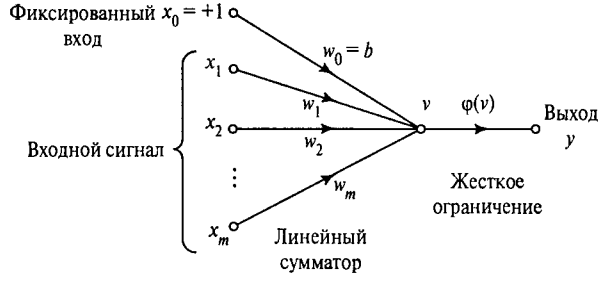


Рис. 3.8. Эквивалентный граф передачи сигнала для персептрона. Зависимость от времени опущена для ясности

Следовательно, выход линейного сумматора можно записать в более компактной форме

$$v(n) = \sum_{i=0}^{m} w_i(n)x_i(n) = \mathbf{w}^T(n)x(n), \tag{3.52}$$

где $w_0(n)$ — пороговое значение $b(n)$. При фиксированном значении n уравнение $\mathbf{w}^T\mathbf{x} = 0$ в m-мерном пространстве с координатами $x_1, x_2, \ldots, x_m$ определяет гиперплоскость (для некоторого предопределенного значения порога), которая является поверхностью решений для двух различных классов входных сигналов.

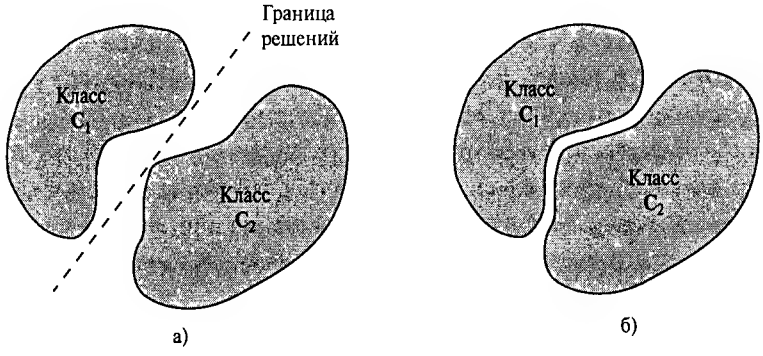


Рис. 3.9. Пара линейно-разделимых (а) и нелинейно-разделимых образов (б)

Чтобы персептрон функционировал корректно, два класса, $\mathbf{C}_1$ и $\mathbf{C}_2$, должны быть *линейно-разделимыми* (linearly separable). Это, в свою очередь, означает, что для правильной классификации образы должны быть значительно отдалены друг от друга, чтобы поверхность решений могла представлять собой гиперплоскость. Это требование проиллюстрировано на рис. 3.9 для случая двумерного персептрона. На рис. 3.9, *а* два класса — $\mathbf{C}_1$ и $\mathbf{C}_2$ — значительно удалены друг от друга, и их можно разделить гиперплоскостью (в данном случае — прямой). Если эти два класса сдвинуть ближе друг к другу (рис. 3.9, *б*), они станут нелинейно-разделимыми. Такая ситуация выходит за рамки вычислительных возможностей персептрона.

Теперь предположим, что входные переменные персептрона принадлежат двум линейно-разделимым классам. Пусть $\mathbf{X}_1$ — подмножество векторов обучения $\mathbf{x}_1(1)$, $\mathbf{x}_1(2)$, ..., которое принадлежит классу $\mathbf{C}_1$, а $\mathbf{X}_2$ — подмножество векторов обучения $\mathbf{x}_2(1)$, $\mathbf{x}_2(2)$, ..., относящееся к классу $\mathbf{C}_2$. Объединение подмножеств $\mathbf{X}_1$ и $\mathbf{X}_2$ составляет все обучающее множество $\mathbf{X}$. Использование подмножеств $\mathbf{X}_1$ и $\mathbf{X}_2$ для обучения классификатора позволит настроить вектор весов $\mathbf{w}$ таким образом, что два класса — $\mathbf{C}_1$ и $\mathbf{C}_2$ — будут линейно-разделимыми. Это значит, что существует такой вектор весовых коэффициентов $\mathbf{w}$, для которого истинно следующее утверждение:

$$\begin{aligned} &\mathbf{w}^T\mathbf{x} > 0 \text{ для любого входного вектора } \mathbf{x}, \text{ принадлежащего классу } \mathbf{C}_1, \\ &\mathbf{w}^T\mathbf{x} \leq 0 \text{ для любого входного вектора } \mathbf{x}, \text{ принадлежащего классу } \mathbf{C}_2. \end{aligned} \quad (3.53)$$

Во второй строке утверждения (3.53) мы произвольно указали, что при равенстве $\mathbf{w}^T\mathbf{x} = 0$ входной вектор $\mathbf{x}$ принадлежит именно классу $\mathbf{C}_2$. При определенных таким образом подмножествах $\mathbf{X}_1$ и $\mathbf{X}_2$ задача обучения элементарного персептрона сводится к нахождению такого вектора весов $\mathbf{w}$, для которого выполняются оба неравенства (3.53).

Алгоритм адаптации вектора весовых коэффициентов элементарного персептрона можно сформулировать следующим образом.

Если n-й элемент $\mathbf{x}(n)$ обучающего множества корректно классифицирован с помощью весовых коэффициентов $\mathbf{w}(n)$, вычисленных на n-м шаге алгоритма, то вектор весов не корректируется, т.е. действует следующее правило:

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) \text{ если } \mathbf{w}^T\mathbf{x}(n) > 0 \text{ и } \mathbf{x}(n) \in \mathbf{C}_1,\\ \mathbf{w}(n+1) &= \mathbf{w}(n) \text{ если } \mathbf{w}^T\mathbf{x}(n) \leq 0 \text{ и } \mathbf{x}(n) \in \mathbf{C}_2.\end{aligned} \tag{3.54}$$

В противном случае вектор весов персептрона подвергается коррекции в соответствии со следующим правилом:

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) - \eta(n)\mathbf{x}(n), \text{ если } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ и } \mathbf{x}(n) \in \mathbf{C}_2,\\ \mathbf{w}(n+1) &= \mathbf{w}(n) + \eta(n)\mathbf{x}(n), \text{ если } \mathbf{w}^T(n)\mathbf{x}(n) \leq 0 \text{ и } \mathbf{x}(n) \in \mathbf{C}_1,\end{aligned} \tag{3.55}$$

где интенсивность настройки вектора весов на шаге n определяется *параметром скорости обучения* $\eta(n)$.

Если $\eta(n) = \eta > 0$, где η — константа, не зависящая от номера итерации n, вышеописанный алгоритм называется *правилом адаптации с фиксированным приращением* (fixed increment adaptation rule).

Ниже мы сначала докажем сходимость правила адаптации с фиксированным приращением для $\eta = 1$. Ясно, что само значение η не играет особой роли, если оно положительно. Значение параметра η, отличное от единицы, обеспечивает масштабирование образов, не влияя на их разделимость. Случай с переменным коэффициентом $\eta(n)$ будет рассмотрен немного позже.

В приведенном доказательстве считается, что в начале процесса обучения вектор весовых коэффициентов равен нулю, $\mathbf{w}(0)$=$\mathbf{0}$. Предположим, что для $n = 1, 2, \ldots$, $\mathbf{w}^T(n)\mathbf{x}(n) < 0$, а входной вектор $\mathbf{x}(n)$ принадлежит подмножеству $\mathbf{X}_1$. Это значит, что персептрон некорректно классифицировал векторы $\mathbf{x}(1)$, $\mathbf{x}(2)$, ..., т.е. условие (3.53) не выполнено. Следовательно, для $\eta(n) = 1$ можно использовать вторую строку правила (3.55):

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mathbf{x}(n), \text{ для } \mathbf{x}(n) \in \mathbf{C}_1. \tag{3.56}$$

Поскольку начальное состояние $\mathbf{w}(0) = \mathbf{0}$, то уравнение (3.56) для $\mathbf{w}(n+1)$ можно решить итеративно и получить следующий результат:

$$\mathbf{w}(n+1) = \mathbf{x}(1) + \mathbf{x}(1) + \ldots + \mathbf{x}(n). \tag{3.57}$$

Так как по предположению классы $\mathbf{C}_1$ и $\mathbf{C}_2$ являются линейно-разделимыми, то существует такое решение $\mathbf{w}_0$, при котором будет выполняться условие $\mathbf{w}^T\mathbf{x}(n) > 0$ для векторов $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(n)$, принадлежащих подмножеству $\mathbf{X}_1$. Для фиксированного решения $\mathbf{w}_0$ можно определить такое положительное число α, что

$$\alpha = \min_{\mathbf{x}(n) \in \mathbf{X}_1} \mathbf{w}_0^T \mathbf{x}(n). \tag{3.58}$$

Умножая обе части уравнения (3.57) на вектор-строку $\mathbf{w}_0^T$, получим:

$$\mathbf{w}_0^T \mathbf{w}(n+1) = \mathbf{w}_0^T \mathbf{x}(1) + \mathbf{w}_0^T \mathbf{x}(2) + \ldots + \mathbf{w}_0^T \mathbf{x}(n).$$

В свете определения (3.58) имеем:

$$\mathbf{w}_0^T \mathbf{w}(n+1) \geq n\alpha. \tag{3.59}$$

Теперь можно использовать *неравенство Гучи–Шварца* (Gauchi-Schwartz inequality). Для двух векторов, $\mathbf{w}_0$ и $\mathbf{w}(n+1)$, его можно записать следующим образом:

$$\|\mathbf{w}_0\|^2 \|\mathbf{w}(n+1)\|^2 \geq \left[\mathbf{w}_0^T \mathbf{w}(n+1)\right]^2, \tag{3.60}$$

где $\|\cdot\|$ — Евклидова норма векторного аргумента; $\mathbf{w}_0^T \mathbf{w}(n+1)$ — скалярное произведение векторов. Заметим, что согласно (3.59) $\left[\mathbf{w}_0^T \mathbf{w}(n+1)\right]^2 \geq n^2\alpha^2$. Учитывая это в (3.60), получим:

$$\|\mathbf{w}_0\|^2 \|\mathbf{w}(n+1)\|^2 \geq n^2\alpha^2$$

или

$$\|\mathbf{w}(n+1)\|^2 \geq \frac{n^2\alpha^2}{\|\mathbf{w}_0\|^2}. \tag{3.61}$$

Теперь подойдем к проблеме с другой стороны. В частности, перепишем уравнение (3.56) в следующем виде:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{x}(k), \text{ для } k = 1, \ldots, n \text{ и } \mathbf{x}(k) \in \mathbf{X}_1. \tag{3.62}$$

Вычисляя Евклидову норму векторов в обеих частях уравнения (3.62), получим:

$$\|\mathbf{w}(k+1)\|^2 = \|\mathbf{w}(k)\|^2 + \|\mathbf{x}(k)\|^2 + 2\mathbf{w}^T(k)\mathbf{x}(k). \tag{3.63}$$

Если персептрон некорректно классифицировал входной вектор $\mathbf{x}(k)$, принадлежащий подмножеству $\mathbf{X}_1$, то $\mathbf{w}^T(k)\mathbf{x}(k) < 0$. Следовательно, из (3.63) получим выражение

$$||\mathbf{w}(k+1)||^2 \leq ||\mathbf{w}(k)||^2 + ||\mathbf{x}(k)||^2$$

или

$$||\mathbf{w}(k+1)||^2 - ||\mathbf{w}(k)||^2 \leq ||\mathbf{x}(k)||^2 \text{ для } k = 1, \ldots, n. \tag{3.64}$$

Применяя эти неравенства последовательно для $k = 1, \ldots, n$ и учитывая изначальное допущение, что $\mathbf{w}(0)=\mathbf{0}$, приходим к неравенству

$$\|\mathbf{w}(n+1)\|^2 \leq \sum_{k=1}^{n} \|\mathbf{x}(k)\|^2 \leq n\beta, \tag{3.65}$$

где β — положительное число, определяемое следующим образом:

$$\beta = \max_{\mathbf{x}(k)\in\mathbf{X}_1} \|\mathbf{x}(k)\|^2. \tag{3.66}$$

Уравнение (3.65) означает, что Евклидова норма вектора весов $\mathbf{w}(n+1)$ линейно возрастает с увеличением номера итерации n.

Результат, описываемый неравенством (3.65), при больших n вступает в противоречие с полученным ранее результатом (3.61). Следовательно, номер итерации n не может превышать некоторого значения n_{max}, при котором неравенства (3.61) и (3.65) удовлетворяются со знаком равенства. Это значит, что число n_{max} должно быть решением уравнения

$$\frac{n_{\max}^2 \alpha^2}{\|\mathbf{w}_0\|^2} = n_{\max}\beta.$$

Разрешая это уравнение для n_{max} относительно $\mathbf{w}_0$, получим:

$$n_{\max} = \frac{\beta \|\mathbf{w}_0\|^2}{\alpha^2}. \tag{3.67}$$

Таким образом, мы доказали, что для $\eta(n) = 1$ и $\mathbf{w}(0)=\mathbf{0}$ в предположении существования вектора решения $\mathbf{w}_0$ процесс адаптации синаптических весов персептрона должен прекращаться не позднее итерации $n_{\max}$. Обратите внимание, что согласно (3.58), (3.66) и (3.67) решение для $\mathbf{w}_0$ и $n_{\max}$ не *единственно*.

Теперь можно сформулировать *теорему сходимости для алгоритма обучения персептрона с фиксированным приращением* (fixed-increment convergence theorem) для персептрона [899].

Пусть подмножества векторов обучения $\mathbf{X}_1$ *и* $\mathbf{X}_2$ *линейно-разделимы. Пусть входные сигналы поступают персептрону только их этих подмножеств. Тогда алгоритм обучения персептрона сходится после некоторого числа* n_0 *итераций в том смысле, что*

$$\mathbf{w}(n_0) = \mathbf{w}(n_0 + 1) = \mathbf{w}(n_0 + 2) = \ldots$$

является вектором решения для $n_0 \leq n_{\max}$.

Теперь рассмотрим *абсолютную процедуру* адаптации однослойного персептрона *на основе коррекции ошибок* (absolute error-correction procedure), в которой $\eta(n)$ — переменная величина. В частности, пусть $\eta(n)$ — наименьшее целое число, для которого выполняется соотношение

$$\eta(n)\mathbf{x}^T(n)\mathbf{x}(n) > |\mathbf{w}^T(n)\mathbf{x}(n)|.$$

Согласно этой процедуре, если скалярное произведение $\mathbf{w}^T(n)\mathbf{x}(n)$ на шаге n имеет неверный знак, то $\mathbf{w}^T(n+1)\mathbf{x}(n)$ на итерации $n+1$ будет иметь правильный знак. Таким образом, предполагается, что если знак произведения $\mathbf{w}^T(n)\mathbf{x}(n)$ некорректен, то можно изменить последовательность обучения для итерации $n+1$, приняв $\mathbf{x}(n+1)$=$\mathbf{x}(n)$. Другими словами, каждый из образов представляется персептрону до тех пор, пока он не будет классифицирован корректно.

Заметим, что использование отличного от нулевого исходного состояния $\mathbf{w}(0)$ приводит к увеличению или уменьшению количества итераций, необходимых для сходимости, в зависимости от того, насколько близким окажется исходное состояние $\mathbf{w}(0)$ к решению $\mathbf{w}_0$. Однако, независимо от исходного значения $\mathbf{w}(0)$, сходимость все равно будет обеспечена.

В табл. 3.2 представлен *алгоритм сходимости персептрона* (perceptron convergence algorithm) [657] в краткой форме. Символ $\mathrm{sgn}(\cdot)$, использованный на третьем шаге для вычисления фактического отклика персептрона, означает *функцию вычисления знака* (сигнум):

$$\mathrm{sgn}(v) = \begin{cases} +1\ , & \text{если } v > 0, \\ -1, & \text{если } v < 0. \end{cases} \tag{3.68}$$

Таким образом, *дискретный отклику*(n) *персептрона* (quantized responce of perceptron) можно выразить в компактной форме:

$$y(n) = \mathrm{sgn}(\mathbf{w}^T(n)\mathbf{x}(n)). \tag{3.69}$$

ТАБЛИЦА 3.2. Алгоритм сходимости персептрона в краткой форме

Переменные и параметры

$\mathbf{x}(n) = [+1, x_1(n), \ldots, x_m(n)]^T$ — вектор-строка размерности $m+1$;

$\mathbf{w}(n) = [b(n), w_1(n), \ldots, w_m(n)]^T$ — вектор-строка размерности $m+1$;

$b(n)$ — порог;

$y(n)$ — фактический отклик (дискретизированный);

$d(n)$ — желаемый отклик;

$0 < \eta \leq 1$ — параметр скорости обучения.

1. *Инициализация*

Пусть $\mathbf{w}(0)=\mathbf{0}$. Последующие вычисления выполняются для шагов $n = 1, 2, \ldots$.

2. *Активация*

На шаге n активируем персептрон, используя вектор $\mathbf{x}(n)$ с вещественными компонентами и желаемый отклик $d(n)$.

3. *Вычисление фактического ответа*

Вычисляем фактический отклик персептрона:

$$y(n) = \operatorname{sgn}(\mathbf{w}^T(n)\mathbf{x}(n)),$$

где $sgn(\cdot)$ — функция вычисления знака аргумента.

4. *Адаптация вектора весов*

Изменяем вектор весов персептрона:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n),$$

где

$$d(n) = \begin{cases} +1, & \text{если } \mathbf{x}(n) \in \mathbf{C}_1, \\ -1, & \text{если } \mathbf{x}(n) \in \mathbf{C}_2. \end{cases}$$

5. *Продолжение*

Увеличиваем номер итерации n на единицу и возвращаемся к п. 2 алгоритма.

Заметим, что входной сигнал $\mathbf{x}(n)$ является вектором-строкой размерности $(m+1)$, первый элемент которого — фиксированная величина (равная $+1$) на всем протяжении алгоритма. Соответственно первым элементом вектора-строки весовых коэффициентов $\mathbf{w}(n)$ размерности $m+1$ является порог $b(n)$. Следует отметить еще одну важную деталь, приведенную в табл. 3.2, — *дискретный желаемый отклик* $d(n)$ определяется выражением

$$d(n) = \begin{cases} +1, & \text{если } \mathbf{x}(n) \in \mathbf{C}_1, \\ -1, & \text{если } \mathbf{x}(n) \in \mathbf{C}_2. \end{cases} \tag{3.70}$$

Таким образом, алгоритм адаптации вектора весовых коэффициентов $\mathbf{w}(n)$ соответствует *правилу обучения на основе коррекции ошибок* (error-correction learning rule)

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n), \tag{3.71}$$

где η — *параметр скорости обучения*, а разность $d(n) - y(n)$ выступает в роли *сигнала ошибки*. Параметр скорости обучения является положительной константой, принадлежащей интервалу $0<\eta \leq 1$. Выбирая значение параметра скорости обучения из этого диапазона, следует учитывать два взаимоисключающих требования [657].

- *Усреднение* (averaging) предыдущих входных сигналов, обеспечивающее устойчивость оценки вектора весов, требует малых значений η.
- *Быстрая адаптация* (fast adaptation) к реальным изменениям распределения процесса, отвечающего за формирование векторов входного сигнала $\mathbf{x}$, требует больших значений η.

3.10. Взаимосвязь персептрона и байесовского классификатора в гауссовой среде

Персептрон имеет определенную связь с классической системой классификации образов, получившей название *байесовского классификатора* (Bayes classifier). В условиях гауссовой среды байесовский классификатор превращается в обычный линейный классификатор. Такую же форму имеет персептрон. Однако линейная природа персептрона не зависит от стохастических свойств среды. В этом разделе речь пойдет о взаимосвязи персептрона и байесовского классификатора, что позволит глубже изучить природу и работу персептрона. Начнем этот раздел с краткого описания байесовского классификатора.

Байесовский классификатор

В *байесовском классификаторе* (Bayes classifier) или *байесовской процедуре проверки гипотез* (hypothesis testing procedure) минимизируется средний риск, который обозначается символом $\mathbf{R}$. Для задачи двух классов ($\mathbf{C}_1$ и $\mathbf{C}_2$) средний риск в [1080] определяется следующим образом:

$$\begin{aligned}\mathbf{R} = {} & c_{11}p_1 \int_{\mathbf{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_1)d\mathbf{x} + c_{22}p_2 \int_{\mathbf{X}_2} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_2)d\mathbf{x} \\ & + c_{21}p_1 \int_{\mathbf{X}_2} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_1)d\mathbf{x} + c_{12}p_2 \int_{\mathbf{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_2)d\mathbf{x},\end{aligned} \tag{3.72}$$

где p_i — *априорная вероятность* (a priori probability) того, что вектор наблюдения $\mathbf{x}$ (представляющий реализацию случайного вектора $\mathbf{X}$) принадлежит подпространству $\mathbf{X}_i$ при $i = 1, 2$, и $p_1 + p_2 = 1$; c_{ij} — стоимость решения в пользу класса $\mathbf{C}_i$, представленного подпространством $\mathbf{X}_i$, когда истинным является класс $\mathbf{C}_j$ (т.е. вектор наблюдения принадлежит подпространству $\mathbf{X}_j$) при $(i, j) = 1, 2$; $f_X(\mathbf{x}|\mathbf{C}_i)$ — функция плотности условной вероятности случайного вектора $\mathbf{X}$ при условии, что вектор наблюдения $\mathbf{x}$ принадлежит подпространству $\mathbf{X}_i$ для $i = 1, 2$.

Первые два слагаемых в правой части уравнения (3.72) представляют *корректные* (correct) решения (т.е. корректные классификации), а вторая пара слагаемых — *некорректные* (incorrect) (т.е. ошибки классификации). Каждое решение взвешивается произведением двух факторов: стоимости принятия решения и относительной частоты его принятия (т.е. априорной вероятности).

Целью является нахождение стратегии *минимизации среднего риска*. В процессе принятия решения каждому вектору наблюдения $\mathbf{x}$ из пространства $\mathbf{X}$ должно быть сопоставлено какое-либо из подпространств — $\mathbf{X}_1$ или $\mathbf{X}_2$. Таким образом,

$$\mathbf{X} = \mathbf{X}_1 + \mathbf{X}_2. \tag{3.73}$$

Соответственно выражение (3.72) можно переписать в эквивалентной форме:

$$\begin{aligned}\mathbf{R} = {} & c_{11}p_1 \int_{\mathbf{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_1)d\mathbf{x} + c_{22}p_2 \int_{\mathbf{X}-\mathbf{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_2)d\mathbf{x} \\ & + c_{21}p_1 \int_{\mathbf{X}-\mathbf{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_1)d\mathbf{x} + c_{12}p_2 \int_{\mathbf{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_2)d\mathbf{x},\end{aligned} \tag{3.74}$$

где $c_{11} < c_{21}$ и $c_{22} < c_{12}$. Принимая во внимание тот факт, что

$$\int_{\mathbf{X}} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_1)d\mathbf{x} = \int_{\mathbf{X}} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_2)d\mathbf{x} = 1, \tag{3.75}$$

уравнение (3.74) можно свести к выражению

$$\begin{aligned}\mathbf{R} = {} & c_{21}p_1 + c_{22}p_2 + \\ & + \int_{\mathbf{X}_1} [p_2(c_{12} - c_{22})f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_2) - p_1(c_{21} - c_{11})f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_1)]\, d\mathbf{x}.\end{aligned} \tag{3.76}$$

Первые два слагаемых в правой части выражения (3.76) представляют собой фиксированную стоимость. Теперь для минимизации среднего риска $\mathbf{R}$ можно вывести следующую стратегию оптимальной классификации.

1. Чтобы интеграл вносил отрицательный вклад в значение риска $\mathbf{R}$, все значения вектора наблюдения $\mathbf{x}$, для которых подынтегральное выражение (т.е. выражение в квадратных скобках) является отрицательным, должны быть отнесены к подпространству $\mathbf{X}_1$ (т.е. к классу $\mathbf{C}_1$).
2. Чтобы интеграл вносил положительный вклад в значение риска $\mathbf{R}$, все значения вектора наблюдения $\mathbf{x}$, для которых подынтегральное выражение является положительным, должны быть исключены из подпространства $\mathbf{X}_1$ (т.е. отнесены к классу $\mathbf{C}_2$).
3. Значения $\mathbf{x}$, при которых подынтегральное выражение равно нулю, не влияют на риск $\mathbf{R}$. Их можно отнести к любому классу произвольным образом. В данном случае будем относить их к подпространству $\mathbf{X}_2$ (т.е. к классу $\mathbf{C}_2$).

Принимая это в расчет, байесовский классификатор можно описать следующим образом.

Если выполняется условие

$$p_1(c_{21} - c_{11}) f_X(\mathbf{x}|\mathbf{C}_1) > p_2(c_{12} - c_{22}) f_X(\mathbf{x}|\mathbf{C}_2),$$

то вектор наблюдения $\mathbf{x}$ *следует относить к подпространству* $\mathbf{X}_1$ *(т.е. к классу* $\mathbf{C}_1$*), в противном случае — к подпространству* $\mathbf{X}_2$ *(т.е. к классу* $\mathbf{C}_2$*).*

Для упрощения изложения введем следующие обозначения:

$$\lambda(\mathbf{x}) = \frac{f_X(\mathbf{x}|\mathbf{C}_1)}{f_X(\mathbf{x}|\mathbf{C}_2)}, \tag{3.77}$$

$$\xi = \frac{p_2(c_{12} - c_{22})}{p_1(c_{21} - c_{11})}. \tag{3.78}$$

Величина $\Lambda(\mathbf{x})$, являющаяся частным двух функций плотности условной вероятности, называется *отношением правдоподобия* (likelihood ratio). Величина ξ называется *пороговым значением* (threshold) процедуры проверки. Заметим, что обе величины — $\Lambda(\mathbf{x})$ и ξ — всегда положительны. В терминах этих величин байесовский классификатор можно переопределить следующим образом.

Если для вектора наблюдения $\mathbf{x}$ *отношение правдоподобия* $\Lambda(\mathbf{x})$ *превышает пороговый уровень* ξ*, то вектор* $\mathbf{x}$ *принадлежит к классу* $\mathbf{C}_1$*, в противном случае — к классу* $\mathbf{C}_2$.

На рис. 3.10, *а* представлена блочная диаграмма байесовского классификатора. Его важные свойства сводятся к следующему.

1. Обработка данных в байесовском классификаторе ограничена исключительно вычислением отношения правдоподобия $\Lambda(\mathbf{x})$.

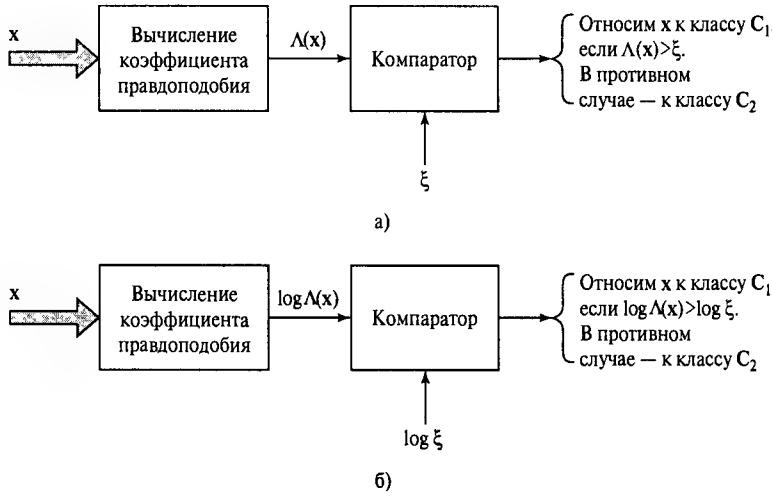


Рис. 3.10. Две эквивалентные реализации байесовского классификатора: на основе отношения правдоподобия (а) и его логарифма (б)

2. Эти вычисления полностью инвариантны по отношению к значениям априорной вероятности и стоимости, назначенным в процессе принятия решения. Эти значения влияют на величину порога ξ.

С вычислительной точки зрения более удобно работать с логарифмом отношения правдоподобия, а не с самим коэффициентом. К этому заключению приходим по двум причинам. Во-первых, логарифм является монотонной функцией. Во-вторых, значения $\Lambda(\mathbf{x})$ и ξ всегда положительны. Исходя из этого, байесовский классификатор можно реализовать в эквивалентной форме, показанной на рис. 3.10, *б*. Такой подход называют *логарифмическим критерием отношения правдоподобия* (log-likelihood ratio test).

Байесовский классификатор и распределение Гаусса

Рассмотрим частный случай задачи классификации на два класса, в котором случайная величина имеет распределение Гаусса. Среднее значение случайного вектора $\mathbf{X}$ зависит от того, какому классу принадлежат его реализации — $\mathbf{C}_1$ или $\mathbf{C}_2$, однако матрица ковариации $\mathbf{X}$ остается одной и той же для обоих классов. Таким образом, можно записать следующее.

Класс $\mathbf{C}_1$:
$$E[\mathbf{X}] = \mu_1,$$
$$E[(\mathbf{X} - \mu_1)(\mathbf{X} - \mu_1)^T] = \mathbf{C}.$$

Класс $\mathbf{C}_2$:
$$E[\mathbf{X}] = \mu_2,$$
$$E[(\mathbf{X} - \mu_2)(\mathbf{X} - \mu_2)^T] = \mathbf{C}.$$

Матрица ковариации $\mathbf{C}$ не является диагональной, а это значит, что образы классов $\mathbf{C}_1$ и $\mathbf{C}_2$ *коррелированы*. Предполагается, что матрица $\mathbf{C}$ является несингулярной, поэтому существует обратная матрица $\mathbf{C}^{-1}$.

Используя эти соглашения, функцию плотности условной вероятности $\mathbf{X}$ можно представить в следующем виде:

$$f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_i) = \frac{1}{(2\pi)^{m/2}(\det(\mathbf{C}))^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)), \quad i = 1,\ 2, \quad (3.79)$$

где m — размерность вектора наблюдения $\mathbf{x}$.

Введем следующие предположения.

1. Вероятность принадлежности образа обоим классам — $\mathbf{C}_1$ и $\mathbf{C}_2$ — одинакова, т.е.

$$p_1 = p_2 = 1/2. \quad (3.80)$$

2. Ошибка классификации имеет постоянную стоимость, а корректная классификация стоимости не имеет:

$$c_{12} = c_{21} \text{и } c_{11} = c_{22} = 0. \quad (3.81)$$

Теперь мы обладаем всей информацией, необходимой для построения байесовского классификатора для двух классов. В частности, подставляя (3.79) в (3.77) и вычисляя натуральный логарифм, после упрощения получим:

$$\begin{aligned}\log\Lambda(\mathbf{x}) &= -1/2(\mathbf{x}-\boldsymbol{\mu}_1)^T\mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + 1/2(\mathbf{x}-\boldsymbol{\mu}_2)^T\mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) = \\ &= (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\mathbf{C}^{-1}\mathbf{x} + 1/2(\boldsymbol{\mu}_2^T\mathbf{C}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\mathbf{C}^{-1}\boldsymbol{\mu}_1).\end{aligned} \quad (3.82)$$

Подставляя (3.80) и (3.81) в (3.78) и находя натуральный логарифм, приходим к соотношению

$$\log\ \xi = 0. \quad (3.83)$$

Выражения (3.82) и (3.83) свидетельствуют о том, что байесовский классификатор для данной задачи является *линейным классификатором*, описываемым соотношением

$$y = \mathbf{w}^T\mathbf{x} + b, \quad (3.84)$$

где

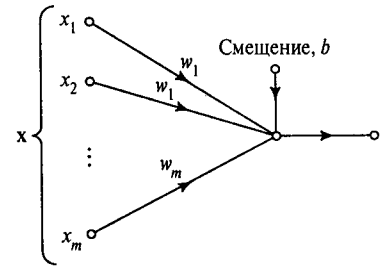


Рис. 3.11. Граф передачи сигнала байесовского классификатора в гауссовой среде

$$y = \log\ \Lambda(\mathbf{x}), \tag{3.85}$$

$$\mathbf{w} = \mathbf{C}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2), \tag{3.86}$$

$$b = \frac{1}{2}(\boldsymbol{\mu}_2^T \mathbf{C}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T \mathbf{C}^{-1}\boldsymbol{\mu}_1). \tag{3.87}$$

Более точно, классификатор представляет собой линейный сумматор с вектором весов $\mathbf{w}$ и порогом b (рис. 3.11).

Теперь с учетом (3.84) логарифмический критерий отношения правдоподобия для задачи классификации на два класса можно описать следующим образом.

Если выходной сигнал y линейного сумматора (содержащего порог b) положителен, вектор наблюдения $\mathbf{x}$ относится к классу $\mathbf{C}_1$, в противном случае — к классу $\mathbf{C}_2$.

Описанный выше байесовский классификатор в гауссовой среде аналогичен персептрону в том смысле, что оба классификатора являются линейными (см. (3.71) и (3.84)). Однако между ними существует ряд мелких и важных различий, на которых следует остановиться [657].

- Персептрон работает при условии, что классифицируемые образы *линейно-разделимы* (linearly separable). Распределение Гаусса в контексте байесовского классификатора *предполагает* их пересечение, что исключает их линейную разделимость. Границы этого пересечения определяется посредством векторов $\boldsymbol{\mu}_1$ и $\boldsymbol{\mu}_2$ и матрицы ковариации $\mathbf{C}$. Природа пересечения проиллюстрирована на рис. 3.12 для частного случая скалярной случайной переменной (т.е. размерность $m = 1$). Если входные сигналы неразделимы и их функции распределения пересекаются так, как показано на рисунке, алгоритм обучения персептрона не сходится, так как границы областей решения могут постоянно смещаться.
- Байесовский классификатор минимизирует вероятность ошибки классификации. Эта минимизация не зависит от пересечения между распределениями Гаусса двух классов. Например, в частном случае, показанном на рис. 3.12, байесовский классификатор всегда будет помещать границу областей решения в точку пересечения функций гауссова распределения классов $\mathbf{C}_1$ и $\mathbf{C}_2$.

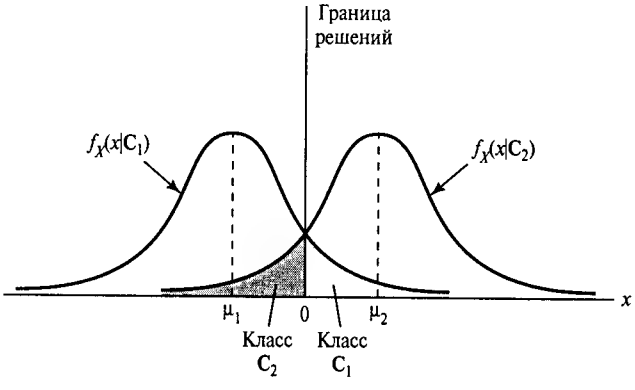


Рис. 3.12. Две пересекающиеся одномерные функции распределения Гаусса

- Алгоритм работы персептрона является *непараметрическим* (nonparametric), т.е. относительно формы рассматриваемых распределений никаких предварительных предположений не делается. Работа алгоритма базируется на коррекции ошибок, возникающих в точках пересечения функций распределения. Таким образом, персептрон хорошо работает с входными сигналами, генерируемыми нелинейными физическими процессами, даже если их распределения не симметричны и не являются гауссовыми. В отличие от персептрона байесовский классификатор является *параметрическим* (parametric). Он предполагает, что распределения случайных величин являются гауссовыми, а это может ограничить область применения классификатора.
- Алгоритм сходимости персептрона является адаптивным и простым для реализации. Его требования к хранению информации ограничиваются множеством синаптических весов и порогов. Архитектура байесовского классификатора является фиксированной; ее можно сделать адаптивной только за счет усиления требований к хранению информации и усложнения вычислений.

3.11. Резюме и обсуждение

Персептрон и адаптивный фильтр на основе алгоритма LMS связаны самым естественным образом, что проявляется в процессе модификации синаптических связей. Более того, они представляют различные реализации *однослойного персептрона, обучаемого на основе коррекции ошибок* (single layer perceptron based on error-correction-learning). Термин "однослойный" здесь используется для того, чтобы подчеркнуть, что в обоих случаях вычислительный слой состоит из единственного нейрона (это отмечено и в названии данной главы). Однако персептрон и алгоритм минимизации среднеквадратической ошибки отличаются друг от друга в некоторых фундаментальных аспектах.

- Алгоритм минимизации среднеквадратической ошибки использует линейный нейрон, в то время как персептрон основан на формальной модели нейрона Мак-Каллока–Питца.
- Процесс обучения персептрона завершается за конечное число итераций. Алгоритм минимизации среднеквадратической ошибки предполагает *непрерывное обучение* (continuous learning), т.е. обучение происходит до тех пор, пока выполняется обработка сигнала. Этот процесс никогда не останавливается.

Модель Мак-Каллока–Питца накладывает жесткие ограничения на форму нелинейности нейрона. Возникает вопрос: "Будет ли персептрон работать лучше, если эти жесткие ограничения заменить сигмоидальной нелинейностью?" Оказывается, что устойчивое принятие решений персептроном не зависит от вида нелинейности нейрона [983], [984]. Таким образом, можно формально утверждать, что при использовании модели нейрона, которая состоит из линейного сумматора и нелинейного элемента, независимо от формы используемой нелинейности однослойный персептрон будет выполнять классификацию образов только для линейно-разделимых классов.

Завершим обсуждение краткой исторической справкой. Персептрон и алгоритм минимизации среднеквадратической ошибки появились приблизительно в одно и то же время — в конце 1950-х годов. Алгоритм минимизации среднеквадратической ошибки достойно выдержал испытание временем. Он хорошо зарекомендовал себя в качестве "рабочей лошадки" адаптивной обработки сигналов благодаря своей эффективности и простоте реализации. Персептрон Розенблатта имеет более интересную историю.

Первая критика персептрона Розенблатта появилась в [746], где утверждалось, что персептрон Розенблатта не способен к обобщению даже в задаче "исключающего ИЛИ" (двоичной четности), не говоря уже о более общих абстракциях. Вычислительные ограничения персептрона Розенблатта были математически обоснованы в знаменитой книге Минского и Пейперта *Персептроны* [744], [745]. После блестящего и в высшей степени подробного математического анализа персептрона Минский и Пейперт доказали, что персептрон в определении Розенблатта внутренне не способен на глобальные обобщения на базе локальных примеров обучения. В последней главе своей книги Минский и Пейперт высказали предположение, что недостатки персептрона Розенблатта остаются в силе и для его вариаций, в частности для многослойных нейронных сетей. Приведем цитату из этой книги (раздел 13.2) [744].

Персептрон достоин изучения вопреки (и даже благодаря) своим ограничениям. Он имеет множество свойств, заслуживающих внимания: линейность, интригующую теорему обучения, образцовую простоту в смысле организации параллельных вычислений. Нет оснований полагать, что некоторые из его преимуществ сохранятся в многослойной версии. Тем не менее мы рассматриваем его как важный объект исследований, для того чтобы обосновать (или отвергнуть) наш интуитивный приговор: расширение персептрона в сторону многослойных систем потенциально безрезультатно.

Это заключение в основном и несет ответственность за возникновение серьезных сомнений в вычислительных возможностях не только персептрона, но и нейронных сетей в целом вплоть до середины 1980-х годов.

Однако история показала, что предположение, высказанное Минским и Пейпертом, было бездоказательным. В настоящее время мы имеем ряд усовершенствованных форм нейронных сетей, которые с вычислительной точки зрения мощнее персептрона Розенблатта. Например, многослойные персептроны, обучаемые с помощью алгоритма обратного распространения ошибки (глава 4), сети на основе радиальных базисных функций (глава 5) и машины опорных векторов (глава 6) преодолевают вычислительные ограничения однослойного персептрона различными способами.

Задачи

Безусловная оптимизация

3.1. Исследуйте метод наискорейшего спуска для единственного весового коэффициента w с помощью следующей функции стоимости:

$$\mathbf{E}(w) = \frac{1}{2}\sigma^2 - r_{xd}w + \frac{1}{2}r_x w^2,$$

где σ^2, r_{xd} и r_x — константы.

3.2. Рассмотрим функцию стоимости

$$\mathbf{E}(\mathbf{w}) = \frac{1}{2}\sigma^2 - \mathbf{r}_{xd}^T\mathbf{w} + \frac{1}{2}\mathbf{w}^T\mathbf{R}_x\mathbf{w},$$

где σ^2 — некоторая константа и

$$\mathbf{r}_{xd} = \begin{bmatrix} 0.8182 \\ 0.354 \end{bmatrix}, \quad \mathbf{R}_x = \begin{bmatrix} 1 & 0.8182 \\ 0.8182 & 1 \end{bmatrix}.$$

а) Найдите оптимальное значение $\mathbf{w}^*$, при котором функция $\mathbf{E}(\mathbf{w})$ достигает своего минимального значения.

б) Используйте метод наискорейшего спуска для вычисления $\mathbf{w}^*$ при следующих значениях параметра скорости обучения: η =0,3, η = 1,0.

Для каждого случая постройте траекторию вектора весов $\mathbf{w}(n)$ в плоскости $\mathbf{W}$.

Примечание. Траектории, полученные для обоих значений параметра скорости обучения, должны соответствовать рис. 3.2.

3.3. Рассмотрите функцию стоимости (3.24), которая является модифицированной формой суммы квадратов ошибок, определяемой соотношением (3.17). Покажите, что применение метода Гаусса–Ньютона к выражению (3.24) приводит к формуле модификации весов (3.23).

Алгоритм LMS

3.4. Матрица корреляции $\mathbf{R}_\mathbf{x}$ входного вектора $\mathbf{x}(n)$ в алгоритме минимизации среднеквадратической ошибки определяется выражением

$$\mathbf{R}_\mathbf{x} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

Определите диапазон значений параметра скорости обучения η алгоритма LMS, при котором он сходится в смысле среднеквадратического значения.

3.5. *Нормализованный алгоритм минимизации среднеквадратической ошибки* описывается следующим рекурсивным выражением для вектора весовых коэффициентов:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\eta}{\|\mathbf{x}(n)\|^2} e(n)\mathbf{x}(n),$$

где η — положительная константа, $||\mathbf{x}(n)||$ — Евклидова норма входного вектора $\mathbf{x}(n)$. Сигнал ошибки определяется соотношением

$$e(n) = d(n) - \hat{\mathbf{w}}^T(n)\mathbf{x}(n),$$

где $d(n)$ — желаемый отклик. Покажите, что для сходимости нормализованного алгоритма в смысле среднеквадратического значения требуется, чтобы $0 < \eta < 2$.

3.6. Алгоритм минимизации среднеквадратической ошибки используется для реализации обобщенной системы подавления боковых лепестков, показанной на рис. 2.16. Задайте уравнения, описывающие работу этой системы, если нейронная сеть состоит из единственного нейрона.

3.7. Рассмотрим систему линейного прогнозирования, входной вектор которой состоит из набора примеров $x(n-1)$, $x(n-2)$, ..., $x(n-m)$, где m – порядок прогнозирования. Используйте алгоритм минимизации среднеквадратической ошибки для прогнозирования оценки $\hat{x}(n)$ входного образа $x(n)$. Опишите рекурсивную функцию, которую можно использовать для вычисления весовых коэффициентов $w_1, w_2, \ldots, w_m$ системы.

3.8. Усредненное по всему множеству значение суммы квадратов ошибок можно использовать в качестве функции стоимости. Оно представляет собой среднеквадратическое значение сигнала ошибки

$$\mathbf{J}(\mathbf{w}) = \frac{1}{2}E[e^2(n)] = \frac{1}{2}E[(d(n) - \mathbf{x}^T(n)\mathbf{w})^2].$$

а) Предполагая, что входной вектор $\mathbf{x}(n)$ и желаемый отклик $d(n)$ поступают из стационарной среды, покажите, что

$$\mathbf{J}(\mathbf{w}) = \frac{1}{2}\sigma_d^2 - \mathbf{r}_{\mathbf{x}d}^T\mathbf{w} + \frac{1}{2}\mathbf{w}^T\mathbf{R}_\mathbf{x}\mathbf{w},$$

где

$$\begin{aligned}\sigma_d^2 &= E[d^2(n)],\\ \mathbf{r}_{\mathbf{x}d} &= E[\mathbf{x}(n)d(n)],\\ \mathbf{R}_\mathbf{x} &= E[\mathbf{x}(n)\mathbf{x}^T(n)].\end{aligned}$$

б) Для этой функции стоимости покажите, что вектор градиента и матрица Гессиана для $\mathbf{J}(\mathbf{w})$ имеют следующий вид:

$$\begin{aligned}\mathbf{g} &= -\mathbf{r}_{\mathbf{x}d} + \mathbf{R}_\mathbf{x}\mathbf{w},\\ \mathbf{H} &= \mathbf{R}_\mathbf{x}.\end{aligned}$$

в) В *алгоритме Ньютона* минимизации среднеквадратической ошибки вместо вектора градиента $\mathbf{g}$ используется его моментальное значение [1144]. Покажите, что этот алгоритм с параметром скорости обучения η описывается выражением

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta\,\mathbf{R}_\mathbf{x}^{-1}\mathbf{x}(n)(d(n) - \mathbf{x}^T(n)\mathbf{w}(n)).$$

Предполагается, что матрица, обратная корреляционной, является положительно определенной и вычислена заранее.

3.9. Вернемся к памяти, основанной на матрице корреляции (см. раздел 2.11). Недостаток этой памяти заключался в том, что при предъявлении ей ключевого образа $\mathbf{x}_j$ фактический отклик $\mathbf{y}$ может быть недостаточно близок (в Евклидовом смысле) к желаемому (запомненному образу) $\mathbf{y}_j$ и ассоциация реализуется некорректно. Этот недостаток обусловлен алгоритмом обучения Хебба, который не предполагает обратной связи выхода с входом. Для преодоления этого недостатка в структуру памяти можно добавить механизм коррекции ошибок, обеспечивающий корректную ассоциацию [49].

Пусть $\hat{\mathbf{M}}(n)$ — матрица памяти, полученная на итерации n процесса обучения на основе коррекции ошибок. Эта матрица обучается на ассоциациях, представленных парами

$$\mathbf{x}_k \rightarrow \mathbf{y}_k, k = 1, 2, \ldots, q.$$

а) Адаптируя алгоритм минимизации среднеквадратической ошибки к данной задаче, покажите, что обновленное значение матрицы памяти должно определяться следующим соотношением:

$$\hat{\mathbf{M}}(n+1) = \hat{\mathbf{M}}(n) + \eta\,[\mathbf{y}_k - \hat{\mathbf{M}}(n)\mathbf{x}_k]\mathbf{x}_k^T,$$

где η — параметр интенсивности обучения.

б) Для частного случая автоассоциации, при котором $\mathbf{y}_k = \mathbf{x}_k$, покажите, что при стремлении количества итераций к бесконечности автоассоциация в памяти происходит идеально, т.е.

$$\mathbf{M}(\infty)\mathbf{x}_k = \mathbf{x}_k, k = 1, 2, \ldots, q.$$

в) Результат, описанный в пункте б), можно рассматривать как *задачу вычисления собственных чисел*. В этом контексте $\mathbf{x}_k$ представляет собой собственный вектор матрицы $\mathbf{M}(\infty)$. А что такое собственные значения матрицы $\mathbf{M}(\infty)$?

3.10. В этой задаче необходимо исследовать влияние порогового значения на число обусловленности матрицы корреляции, а значит, и на производительность алгоритма LMS.

Рассмотрим случайный вектор $\mathbf{X}$ с матрицей ковариации

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

и вектором ожидания

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}.$$

а) Вычислите число обусловленности матрицы ковариации **C**.

б) Вычислите число обусловленности матрицы корреляции **R**.

Прокомментируйте влияние порогового значения μ на производительность алгоритма LMS.

Персептрон Розенблатта

3.11. В этой задаче рассматривается еще один метод вывода алгоритма обновления весовых коэффициентов для персептрона Розенблатта. Определим *критерий качества для персептрона* [269] в виде

$$\mathbf{J}_p(\mathbf{w}) = \sum_{\mathbf{x} \in \mathbf{X}(\mathbf{w})}^{(-\mathbf{w}^T \mathbf{x})},$$

где $\mathbf{X}(\mathbf{w})$ — множество примеров, ошибочно классифицированных при данном векторе весов $\mathbf{w}$. Обратите внимание, что при отсутствии ошибочно классифицированных примеров значение функции $\mathbf{J}_p(\mathbf{w})$ равно нулю и выходной сигнал ошибочно классифицируется, если $\mathbf{w}_{\mathbf{x}}^T \leq 0$.

а) Проиллюстрируйте геометрически, что значение $\mathbf{J}_p(\mathbf{w})$ пропорционально сумме Евклидовых расстояний от ошибочно классифицированных примеров до границы областей решений.

б) Определите градиент функции $\mathbf{J}_p(\mathbf{w})$ относительно вектора весов $\mathbf{w}$.

в) Используя полученный в пункте б) результат, покажите, что алгоритм обновления весов персептрона можно записать в следующем виде:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\,(n) \sum_{\mathbf{x} \in \mathbf{X}(\mathbf{w}(n))} \mathbf{x},$$

где $\mathbf{X}(\mathbf{w}(n))$ — множество примеров, ошибочно классифицированных при заданном векторе весовых коэффициентов $\mathbf{w}(n)$; $\eta(n)$ — параметр скорости обучения. Покажите, что для случая коррекции на одном примере этот результат практически совпадает с выражениями (3.54) и (3.55).

3.12. Покажите, что выражения (3.68)–(3.71), отражающие алгоритм сходимости персептрона, согласуются с соотношениями (3.54) и (3.55).

3.13. Рассмотрите два одномерных класса, $\mathbf{C}_1$ и $\mathbf{C}_2$, с гауссовым распределением, дисперсия которого равна 1. Их средние значения соответственно равны

$$\mu_1 = -10,$$
$$\mu_2 = +10.$$

Эти классы являются линейно-разделимыми. Постройте классификатор, разделяющий эти два класса.

3.14. Предположим, что на графе передачи сигнала персептрона (см. рис. 3.6) строгая пороговая функция заменена сигмоидальной

$$\varphi(v) = \tanh\left(\frac{v}{2}\right),$$

где v — индуцированное локальное поле. Классификация, выполняемая персептроном, описывается следующим образом.

Вектор наблюдения $\mathbf{x}$ *принадлежит к классу* $\mathbf{C}_1$, *если выходной сигнал* $y > \theta$, *где* θ — *порог (threshold). В противном случае* $\mathbf{x}$ *принадлежит к классу* $\mathbf{C}_2$.

Покажите, что построенная таким образом поверхность решений является гиперплоскостью.

3.15. а) Персептрон можно использовать для выполнения многих логических функций. Опишите перспетронную реализацию функций логического И, логического ИЛИ и дополнения.

б) Основным ограничением персептрона является его неспособность реализовать логическую функцию исключающего ИЛИ. Объясните почему.

3.16. Уравнения (3.86) и (3.87) определяют вектор весовых коэффициентов и пороговое значение байесовского классификатора для гауссовой среды. Модифицируйте этот классификатор для случая, когда матрица ковариации определяется выражением

$$\mathbf{C} = \sigma^2\mathbf{I},$$

где σ^2 — константа.

Многослойный персептрон

4.1. Введение

Эта глава посвящена важному классу нейронных сетей — многослойным сетям прямого распространения. Обычно сеть состоит из множества сенсорных элементов (входных узлов или узлов источника), которые образуют *входной слой*; одного или нескольких *скрытых слоев* (hidden layer) вычислительных нейронов и одного *выходного слоя* (output layer) нейронов. Входной сигнал распространяется по сети в прямом направлении, от слоя к слою. Такие сети обычно называют *многослойными персептронами* (multilayer percetron). Они представляют собой обобщение однослойного персептрона, рассмотренного в главе 3.

Многослойные персептроны успешно применяются для решения разнообразных сложных задач. При этом обучение с учителем выполняется с помощью такого популярного алгоритма, как *алгоритм обратного распространения ошибки* (error back-propagation algorithm). Этот алгоритм основывается на *коррекции ошибок* (error-correction learning rule). Его можно рассматривать как обобщение столь же популярного алгоритма адаптивной фильтрации — вездесущего алгоритма минимизации среднеквадратической ошибки (LMS), описанного в главе 3 для частного случая отдельного линейного нейрона.

Обучение методом обратного распространения ошибки предполагает два прохода по всем слоям сети: прямого и обратного. При *прямом проходе* (forward pass) образ (входной вектор) подается на сенсорные узлы сети, после чего распространятся по сети от слоя к слою. В результате генерируется набор выходных сигналов, который и является фактической реакцией сети на данный входной образ. Во время прямого прохода все синаптические веса сети *фиксированы*. Во время *обратного прохода* (backward pass) все синаптические веса *настраиваются* в соответствии с правилом коррекции ошибок, а именно: фактический выход сети вычитается из желаемого (целевого) отклика, в результате чего формируется *сигнал ошибки* (error signal). Этот сигнал впоследствии распространяется по сети в направлении, обратном направлению синаптических связей. Отсюда и название — алгоритм обратного распространения ошибки. Синаптические веса настраиваются с целью максимального приближения выходного

сигнала сети к желаемому в статистическом смысле. Алгоритм обратного распространения ошибки в литературе иногда называют упрощенно — *алгоритмом обратного распространения* (back-propagation algorithm). Это название мы и будем использовать в настоящей главе. Процесс обучения, реализуемый этим алгоритмом, называется *обучением на основе обратного распространения* (back-propagation learning).

Многослойные персептроны имеют три отличительных признака.

1. Каждый нейрон сети имеет *нелинейную функцию активации* (nonlinear activation function). Важно подчеркнуть, что данная нелинейная функция является гладкой (т.е. всюду дифференцируемой), в отличие от жесткой пороговой функции, используемой в персептроне Розенблатта. Самой популярной формой функции, удовлетворяющей этому требованию, является *сигмоидальная*[1] (sigmoidal nonlinearity), определяемая *логистической функцией* (logistic function)

$$y_j = \frac{1}{1 + \exp(-v_j)},$$

где v_j — индуцированное локальное поле (т.е. взвешенная сумма всех синаптических входов плюс пороговое значение) нейрона j; y_j — выход нейрона. Наличие нелинейности играет очень важную роль, так как в противном случае отображение "вход-выход" сети можно свести к обычному однослойному персептрону. Более того, использование логистической функции мотивировано биологически, так как в ней учитывается восстановительная фаза реального нейрона.

2. Сеть содержит один или несколько слоев *скрытых нейронов*, не являющихся частью входа или выхода сети. Эти нейроны позволяют сети обучаться решению сложных задач, последовательно извлекая наиболее важные признаки из входного образа (вектора).
3. Сеть обладает высокой степенью *связности* (connectivity), реализуемой посредством синаптических соединений. Изменение уровня связности сети требует изменения множества синаптических соединений или их весовых коэффициентов.

Комбинация всех этих свойств наряду со способностью к обучению на собственном опыте обеспечивает вычислительную мощность многослойного персептрона. Однако эти же качества являются причиной неполноты современных знаний о поведении такого рода сетей. Во-первых, распределенная форма нелинейности и высокая связ-

---

[1] Сигмоидальные функции получили свое названиие благодаря форме своего графика (в виде буквы S). В [727] исследованы два класса сигмоид.

Простые сигмоиды. Произвольные асимптотически ограниченные и строго монотонные функции одной переменной.

Гиперболические сигмоиды. Полное подмиожество простых сигмоид, являющихся обобщением функции гиперболического тангенса.

ность сети существенно усложняют теоретический анализ многослойного персептрона. Во-вторых, наличие скрытых нейронов делает процесс обучения более трудным для визуализации. Именно в процессе обучения необходимо определить, какие признаки входного сигнала следует представлять скрытыми нейронами. Тогда процесс обучения становится еще более сложным, поскольку поиск должен выполняться в очень широкой области возможных функций, а выбор должен производиться среди альтернативных представлений входных образов [458].

Термин "обратное распространение" активно используется после 1986 года, когда он был популяризован в известной книге [912]. Более подробные исторические сведения об алгоритме обратного распространения приводятся в разделе 1.9.

Появление алгоритма обратного распространения стало знаковым событием в области развития нейронных сетей, так как он реализует *вычислительно эффективный* (computationally efficient) метод обучения многослойного персептрона. Было бы слишком самоуверенно утверждать, что алгоритм обратного распространения предлагает действительно оптимальное решение всех потенциально разрешимым проблем, однако он развеял пессимизм относительно обучения многослойных машин, воцарившийся в результате публикации [745].

Структура главы

В данной главе рассматриваются основные аспекты работы многослойного персептрона, а также его обучение методом обратного распространения. Эта глава разбита на семь частей. В первой части (разделы 4.2–4.6) мы обсудим вопросы, связанные с обучением по методу обратного распространения. Начнем с раздела 4.2, подготавливающего почву для дальнейшего изложения этого вопроса. В разделе 4.3 будет представлено детальное описание алгоритма в виде последовательности правил вычисления (chain rule of calculus). Алгоритм в сжатом виде приводится в разделе 4.4. В разделе 4.5 мы продемонстрируем использование алгоритма обратного распространения на примере задачи исключающего ИЛИ (XOR), которая неразрешима с точки зрения однослойного персептрона. В разделе 4.6 приводятся некоторые эвристические и практические рекомендации по повышению производительности алгоритма обратного распространения.

Вторая часть (разделы 4.7–4.9) посвящена использованию многослойного персептрона для распознавания образов. В разделе 4.7 приводится решение статистической задачи распознавания образов с помощью многослойного персептрона. В разделе 4.8 описывается компьютерный эксперимент, иллюстрирующий применение алгоритма обратного распространения в задаче разделения двух перекрывающихся классов с двумерным гауссовым распределением. Значение скрытых нейронов, как детекторов признаков, обсуждается в разделе 4.9.

В третьей части (разделы 4.10–4.11) речь пойдет об исследовании поверхности ошибки. В разделе 4.10 мы обсудим фундаментальную роль обучения по методу обратного распространения в вычислении частных производных аппроксимируемых функций. В разделе 4.11 рассматриваются вычислительные вопросы, связанные с нахождением матрицы Гессе для поверхности ошибки.

В четвертой части (разделы 4.12–4.15) рассматриваются различные вопросы, связанные с производительностью многослойного персептрона, обучаемого с помощью алгоритма обратного распространения. В разделе 4.12 мы обсудим вопросы обобщения, составляющие квинтэссенцию обучения. В разделе 4.13 описывается задача аппроксимации непрерывных функций с помощью многослойного персептрона. Использование перекрестной проверки (cross-validation) в качестве статистического средства проверки достоверности результата обсуждается в разделе 4.14. В разделе 4.15 обсуждаются процедуры снижения порядка многослойного персептрона без потери его производительности. Такие процедуры упрощения сети особенно важны в тех задачах, в которых критическим вопросом является вычислительная сложность.

Пятая часть (разделы 4.16–4.17) завершает исследование алгоритма обратного распространения. В разделе 4.16 анализируются преимущества и ограничения метода обратного распространения. В разделе 4.17 приводятся эвристические рекомендации, ускоряющие сходимость обучения методом обратного распространения.

В шестой части (раздел 4.18) процесс обучения рассматривается под несколько иным углом зрения. Для повышения эффективности обучения с учителем этот процесс рассматривается как численная задача оптимизации. В частности, для реализации обучения с учителем предлагается использовать метод сопряженных градиентов (conjugate-gradient) и квазиньютоновские методы.

В последней части главы (раздел 4.19) основное внимание уделяется самому многослойному персептрону. Здесь будет описана довольно интересная структура сети – *многослойный персептрон свертки* (convolutional multilayer perceptron). Такая сеть успешно использовалась для решения сложных задач распознавания образов.

В завершение главы в разделе 4.20 приводятся общие выводы.

4.2. Вводные замечания

На рис. 4.1 показан архитектурный граф многослойного персептрона с двумя скрытыми слоями и одним выходным слоем. Показанная на рисунке сеть является *полносвязной* (fully connected), что характерно для многослойного персептрона общего вида. Это значит, что каждый нейрон в любом слое сети связан со всеми нейронами (узлами) предыдущего слоя. Сигнал передается по сети исключительно в прямом направлении, слева направо, от слоя к слою.

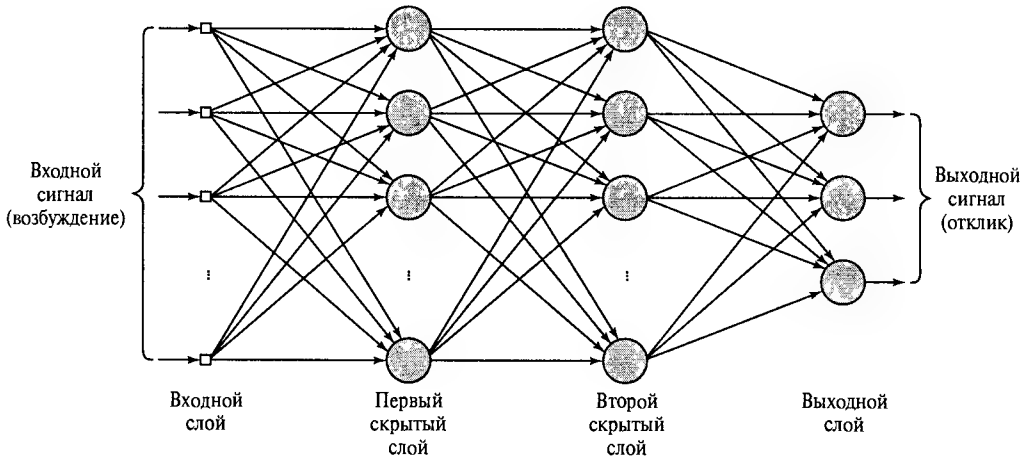


Рис. 4.1. Архитектурный граф многослойного персептрона с двумя скрытыми слоями

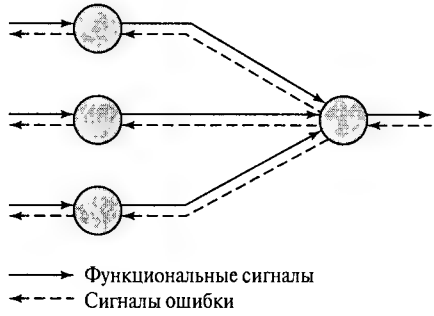


Рис. 4.2. Направление двух основных потоков сигнала для многослойного персептрона: прямое распространение функционального сигнала и обратное распространение сигнала ошибки

На рис. 4.2 показан фрагмент многослойного персептрона. Для этого типа сети выделяют два типа сигналов [816].

1. *Функциональный сигнал* (function signal). Это входной сигнал (стимул), поступающий в сеть и передаваемый вперед от нейрона к нейрону по всей сети. Такой сигнал достигает конца сети в виде выходного сигнала. Будем называть этот сигнал *функциональным* по двум причинам. Во-первых, он предназначен для выполнения некоторой функции на выходе сети. Во-вторых, в каждом нейроне, через который передается этот сигнал, вычисляется некоторая *функция* с учетом весовых коэффициентов. Функциональный сигнал еще называют.
2. *Сигнал ошибки* (error signal). Сигнал ошибки берет свое начало на выходе сети и распространяется в обратном направлении (от слоя к слою). Он получил свое название благодаря тому, что вычисляется каждым нейроном сети на основе функции ошибки, представленной в той или иной форме.

Выходные нейроны (вычислительные узлы) составляют выходной слой сети. Остальные нейроны (вычислительные узлы) относятся к скрытым слоям. Таким образом, скрытые узлы не являются частью входа или выхода сети — отсюда они и получили свое название. Первый скрытый слой получает данные из входного слоя, составленного из сенсорных элементов (входных узлов). Результирующий сигнал первого скрытого слоя, в свою очередь, поступает на следующий скрытый слой, и т.д., до самого конца сети.

Любой скрытый или выходной нейрон многослойного персептрона может выполнять два типа вычислений.

1. Вычисление функционального сигнала на выходе нейрона, реализуемое в виде непрерывной нелинейной функции от входного сигнала и синаптических весов, связанных с данным нейроном.
2. Вычисление оценки вектора градиента (т.е. градиента поверхности ошибки по синаптическим весам, связанным со входами данного нейрона), необходимого для обратного прохода через сеть.

Вывод алгоритма обратного распространения слишком громоздок. Для того чтобы упростить понимание математических выкладок, сначала приведем полный список используемых обозначений.

Обозначения

- Индексы i, j и k относятся к различным нейронам сети. Когда сигнал проходит по сети слева направо, считается, что нейрон j находится на один слой правее нейрона i, а нейрон k — еще на один слой правее нейрона j, если последний принадлежит скрытому слою.
- Итерация (такт времени) n соответствует n-му обучающему образу (примеру), поданному на вход сети.
- Символ $\mathbf{E}(n)$ означает текущую сумму квадратов ошибок (или энергию ошибки) на итерации n. Среднее значение $\mathbf{E}(n)$ по всем значениям n(т.е. по всему обучающему множеству) называется средней энергией ошибки $\mathbf{E}_{\text{av}}$.
- Символ $e_j(n)$ описывает сигнал ошибки на выходе нейрона j на итерации n.
- Символ $d_j(n)$ означает желаемый отклик нейрона j и используется для вычисления $e_j(n)$.
- Символ $y_j(n)$ описывает функциональный сигнал, генерируемый на выходе нейрона j на итерации n.
- Символ $w_{ji}(n)$ используется для обозначения синаптического веса, связывающего выход нейрона i со входом нейрона j на итерации n. Коррекция, применяемая к этому весу на шаге n, обозначается $\Delta w_{ji}(n)$.

- Индуцированное локальное поле (т.е. взвешенная сумма всех синаптических входов плюс порог) нейрона j на итерации n обозначается символом $v_j(n)$. Это значение передается в функцию активации, связанную с нейроном j.
- Функция активации, соответствующая нейрону j и описывающая нелинейную взаимосвязь входного и выходного сигналов этого нейрона, обозначается $\varphi_j(\cdot)$.
- Порог, применяемый к нейрону j, обозначается символом b_j. Его влияние представлено синапсом с весом $w_{j0} = b_j$, соединенным с фиксированным входным сигналом, равным +1.
- i-й элемент входного вектора обозначается $x_i(n)$.
- k-й элемент выходного вектора (образа) обозначается $o_k(n)$.
- Параметр скорости (интенсивности) обучения обозначается символом η.
- Символ m_l обозначает размерность (количество узлов) слоя l многослойного персептрона; $l = 1, 2, \ldots, L$, где L — "глубина" сети. Таким образом, символ m_0 означает размерность входного слоя; m_1 — первого скрытого слоя; m_L — выходного слоя. Для описания размерности выходного слоя будет также использоваться символ M.

4.3. Алгоритм обратного распространения

Сигнал ошибки выходного нейрона j на итерации n (соответствующей n-му примеру обучения) определяется соотношением

$$e_j(n) = d_j(n) - y_j(n). \tag{4.1}$$

Текущее значение энергии ошибки нейрона j определим как $\frac{1}{2}e_j^2(n)$. Соответственно текущее значение $\mathbf{E}(n)$ общей энергии ошибки сети вычисляется путем сложения величин $\frac{1}{2}e_j^2(n)$ по *всем нейронам выходного слоя*. Это "видимые" нейроны, для которых сигнал ошибки может быть вычислен непосредственно. Таким образом, можно записать

$$\mathbf{E}(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n), \tag{4.2}$$

где множество C включает все нейроны выходного слоя сети. Пусть N — общее число образов в обучающем множестве (т.е. мощность этого множества). *Энергия среднеквадратической ошибки* в таком случае вычисляется как нормализованная по N сумма всех значений энергии ошибки $\mathbf{E}(n)$:

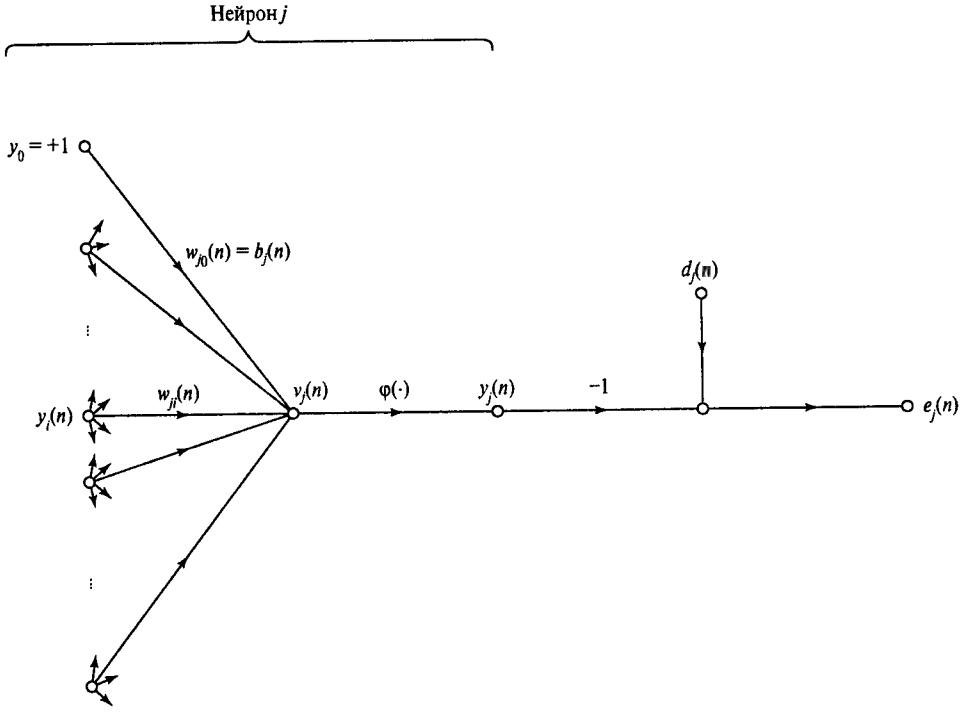


Рис. 4.3. Граф передачи сигнала в пределах некоторого нейрона j

$$\mathbf{E}_{\text{av}}(n) = \frac{1}{N}\sum_{n=1}^{N}\mathbf{E}(n). \tag{4.3}$$

Текущая энергия ошибки $\mathbf{E}(n)$, а значит и средняя энергия ошибки $\mathbf{E}_{\text{av}}$, являются функциями всех свободных параметров (т.е. синаптических весов и значений порога) сети. Для данного обучающего множества энергия $\mathbf{E}_{\text{av}}$ представляет собой *функцию стоимости* (cost function) — меру эффективности обучения. Целью процесса обучения является настройка свободных параметров сети с целью минимизации величины $\mathbf{E}_{\text{av}}$. В процессе минимизации будем использовать аппроксимацию, аналогичную применяемой для вывода алгоритма LMS в главе 3. В частности, рассмотрим простой метод обучения, в котором веса обновляются *для каждого обучающего примера* в пределах одной *эпохи* (epoch), т.е. всего обучающего множества. Настройка весов выполняется в соответствии с ошибками, вычисленными для *каждого* образа, представленного в сети.

Арифметическое среднее отдельных изменений для весов сети по обучающему множеству может служить *оценкой* реальных изменений, произошедших в процессе минимизации функции стоимости $\mathbf{E}_{\text{av}}$ на множестве обучения. Качество этой оценки обсудим немного позже.

Теперь рассмотрим рис. 4.3. На нем изображен нейрон j, на который поступает поток сигналов от нейронов, расположенных в предыдущем слое. Индуцированное локальное поле v_j(n), полученное на входе функции активации, связанной с данным нейроном, равно

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n), \tag{4.4}$$

где m — общее число входов (за исключением порога) нейрона j. Синаптический вес w_{j0} (соответствующий фиксированному входу $y_0 = +1$) равен порогу b_j, применяемому к нейрону j. Функциональный сигнал $y_j(n)$ на выходе нейрона j на итерации n равен

$$y_j(n) = \varphi_j(v_j(n)). \tag{4.5}$$

Аналогично алгоритму LMS, алгоритм обратного распространения состоит в применении к синаптическому весу $w_{ji}(n)$ коррекции $\Delta w_{ji}(n)$, пропорциональной частной производной $\partial \mathbf{E}(n)/\partial w_{ji}(n)$. В соответствии с *правилом цепочки*, или *цепным правилом* (chain rule), градиент можно представить следующим образом:

$$\frac{\partial \mathbf{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathbf{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \tag{4.6}$$

Частная производная $\partial \mathbf{E}(n)/\partial w_{ji}(n)$ представляет собой *фактор чувствительности* (sensitivity factor), определяющий направление поиска в пространстве весов для синаптического веса $w_{ji}(n)$.

Дифференцируя обе части уравнения (4.2) по $e_j(n)$, получим:

$$\frac{\partial \mathbf{E}(n)}{\partial e_j(n)} = e_j(n). \tag{4.7}$$

Дифференцируя обе части уравнения (4.1) по $y_j(n)$, получим соотношение

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1. \tag{4.8}$$

Затем, дифференцируя (4.5) по $v_j(n)$, получим:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)), \tag{4.9}$$

где штрих справа от имени функции обозначает дифференцирование по аргументу.

И, наконец, дифференцируя (4.4) по $w_{ji}(n)$, получим выражение

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_j(n). \tag{4.10}$$

Подставляя результаты (4.7)–(4.10) в выражение (4.6), окончательно получим:

$$\frac{\partial \mathbf{E}(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'_j(v_j(n))y_i(n). \tag{4.11}$$

Коррекция $\Delta w_{ji}(n)$, применяемая к $w_{ji}(n)$, определяется согласно *дельта-правилу* (delta rule):

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathbf{E}(n)}{\partial w_{ji}(n)}, \tag{4.12}$$

где η — *параметр скорости обучения* (learning-rate parameter) алгоритма обратного распространения. Использование знака "минус" в (4.12) связано с реализацией *градиентного спуска* (gradient descent) в пространстве весов (т.е. поиском направления изменения весов, уменьшающего значение энергии ошибки $\mathbf{E}(n)$). Следовательно, подставляя (4.11) в (4.12), получим:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n), \tag{4.13}$$

где *локальный градиент* (local gradient) $\delta_j(n)$ определяется выражением

$$\delta_j(n) = -\frac{\partial \mathbf{E}(n)}{\partial v_j(n)} = -\frac{\partial \mathbf{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi'_j(v_j(n)). \tag{4.14}$$

Локальный градиент указывает на требуемое изменение синаптического веса. В соответствии с (4.14) локальный градиент $\delta_j(n)$ выходного нейрона j равен произведению соответствующего сигнала ошибки $e_j(n)$ этого нейрона и производной $\varphi'_j(v_j(n))$ соответствующей функции активации.

Из выражений (4.13) и (4.14) видно, что ключевым фактором в вычислении величины коррекции $\Delta w_{ji}(n)$ весовых коэффициентов является сигнал ошибки $e_j(n)$ нейрона j. В этом контексте можно выделить два различных случая, определяемых положением нейрона j в сети. В первом случае нейрон j является выходным узлом. Это довольно простой случай, так как для каждого выходного узла сети известен соответствующий желаемый отклик. Следовательно, вычислить сигнал ошибки можно с помощью простой арифметической операции. Во втором случае нейрон j является скрытым узлом. Однако даже если скрытый нейрон непосредственно недоступен, он несет ответственность за ошибку, получаемую на выходе сети. Вопрос состоит в

том, как персонифицировать вклад (положительный или отрицательный) отдельных скрытых нейронов в общую ошибку. Эта проблема называется *задачей назначения коэффициентов доверия* (credit assignment problem) и уже упоминалась в разделе 2.7. Она очень элегантно решается с помощью метода обратного распространения сигнала ошибки по сети.

Случай 1. Нейрон j — выходной узел

Если нейрон j расположен в выходном слое сети, для него известен соответствующий желаемый отклик. Значит, с помощью выражения (4.1) можно определить сигнал ошибки $e_j(n)$ (см. рис. 4.3) и без проблем вычислить градиент $\delta_j(n)$ по формуле (4.14).

Случай 2. Нейрон j — скрытый узел

Если нейрон j расположен в скрытом слое сети, желаемый отклик для него неизвестен. Следовательно, сигнал ошибки скрытого нейрона должен рекурсивно вычисляться на основе сигналов ошибки всех нейронов, с которым он непосредственно связан. Именно здесь алгоритм обратного распространения сталкивается с наибольшей сложностью. Рассмотрим ситуацию, изображенную на рис. 4.4, где изображен скрытый нейрон j. Согласно (4.14), локальный градиент $\delta_j(n)$ скрытого нейрона j можно переопределить следующим образом:

$$\delta_j(n) = -\frac{\partial \mathbf{E}(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \mathbf{E}(n)}{\partial y_j(n)}\varphi_j'(v_j(n)), \tag{4.15}$$

где j — скрытый нейрон, а для получения второго результата использовалась формула (4.9). Для вычисления частной производной $\partial \mathbf{E}(n)/\partial y_j(n)$ выполним некоторые преобразования. На рис. 4.4 видно, что

$$\mathbf{E}(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n), \tag{4.16}$$

где k — выходной нейрон. Соотношение (4.16) — это выражение (4.2), в котором индекс j заменен индексом k. Это сделано для того, чтобы избежать путаницы с индексом j, использованным ранее для скрытого нейрона. Дифференцируя (4.16) по функциональному сигналу $y_j(n)$, получим:

$$\frac{\partial \mathbf{E}(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)}. \tag{4.17}$$

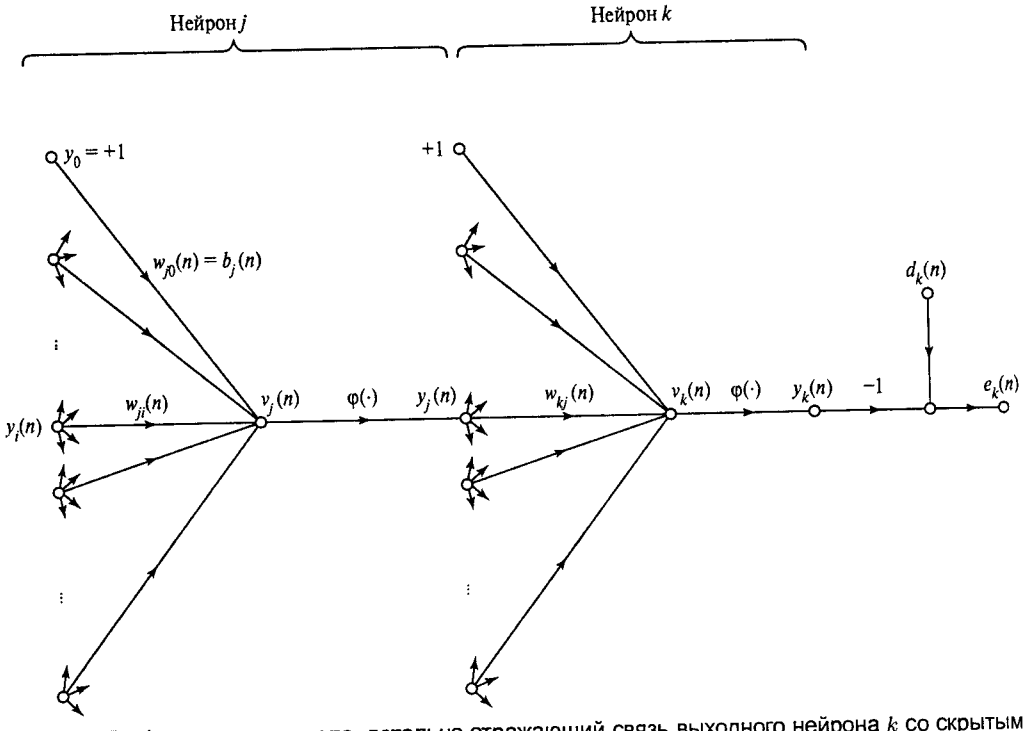


Рис. 4.4. Граф передачи сигнала, детально отражающий связь выходного нейрона k со скрытым нейроном j

Теперь к частной производной $\partial e_k(n)/\partial y_i(n)$ можно применить цепное правило и переписать (4.17) в эквивалентной форме:

$$\frac{\partial \mathbf{E}(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}. \tag{4.18}$$

Однако на рис. 4.4 видно, что для выходного нейрона k

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_j(v_k(n)). \tag{4.19}$$

Отсюда

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)). \tag{4.20}$$

На рис. 4.4 также очевидно, что индуцированное локальное поле нейрона k составляет

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n), \tag{4.21}$$

где m — общее число входов (за исключением порога) нейрона k. Здесь синаптический вес $w_{k0}(n)$ равен порогу b_k, приходящемуся на нейрон k, а соответствующий вход имеет фиксированное значение $+1$. Дифференцируя (4.21) по $y_j(n)$, получим:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n). \tag{4.22}$$

Подставляя (4.20) и (4.22) в (4.18), получим соотношение

$$\frac{\partial \mathbf{E}(n)}{\partial y_j(n)} = -\sum_k e_k(n) \varphi_k'(v_k(n)) w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n), \tag{4.23}$$

где для получения второго результата использовано определение локального градиента (4.14), в котором индекс j заменен индексом k по уже упомянутым соображениям.

В заключение, подставляя выражение (4.23) в (4.15), получим *формулу обратного распространения* (back-propagation formula) для локального градиента $\delta_j(n)$ скрытого нейрона j:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n). \tag{4.24}$$

На рис. 4.5 графически представлена формула (4.24) в предположении, что выходной слой состоит из m_L нейронов.

Множитель $\varphi_j'(v_j(n))$, использованный в формуле (4.24) для вычисления локального градиента $\delta_j(n)$, зависит исключительно от функции активации, связанной со скрытым нейроном j. Второй множитель формулы (сумма по k) зависит от двух множеств параметров. Первое из них — $\delta_k(n)$ — требует знания сигналов ошибки $e_k(n)$ для всех нейронов слоя, находящегося правее скрытого нейрона j, которые непосредственно связаны с нейроном j (см. рис. 4.4). Второе множество — $w_{kj}(n)$ — состоит из синаптических весов этих связей.

Теперь можно свести воедино все соотношения, которые мы вывели для алгоритма обратного распространения. Во-первых, коррекция $\Delta w_{ji}(n)$, применяемая к синаптическому весу, соединяющему нейроны i и j, определяется следующим дельта-правилом:

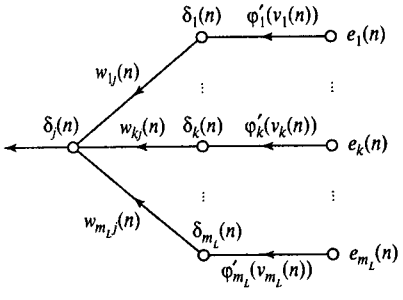


Рис. 4.5. Граф передачи сигнала для фрагмента системы, задействованного в обратном распространении сигналов ошибки

$$\begin{pmatrix}\text{Коррекция}\\ \text{веса}\\ \Delta w_{ji}(n)\end{pmatrix} = \begin{pmatrix}\text{Параметр ско-}\\ \text{рости обучения}\\ \eta\end{pmatrix} \cdot \begin{pmatrix}\text{Локальный}\\ \text{градиент}\\ \delta_j(n)\end{pmatrix} \cdot \begin{pmatrix}\text{Входной сиг-}\\ \text{нал нейрона } j\\ y_i(n)\end{pmatrix} \quad (4.25)$$

Во-вторых, значение локального градиента $\delta_j(n)$ зависит от положения нейрона в сети.

1. Если нейрон j — выходной, то градиент $\delta_j(n)$ равен произведению производной $\varphi_j'(v_j(n))$ на сигнал ошибки $e_j(n)$ для нейрона j (см. выражение (4.14)).
2. Если нейрон j — скрытый, то градиент $\delta_j(n)$ равен произведению производной $\varphi_j'(v_j(n))$ на взвешенную сумму градиентов, вычисленных для нейронов следующего скрытого или выходного слоя, которые непосредственно связаны с данным нейроном j (см. выражение (4.24)).

Два прохода вычислений

При использовании алгоритма обратного распространения различают два прохода, выполняемых в процессе вычислений. Первый проход называется прямым, второй — обратным.

При *прямом проходе* (forward pass) синаптические веса остаются неизменными во всей сети, а функциональные сигналы вычисляются последовательно, от нейрона к нейрону. Функциональный сигнал на выходе нейрона j вычисляется по формуле

$$y_j(n) = \varphi(v_j(n)), \quad (4.26)$$

где $v_j(n)$ — индуцированное локальное поле нейрона j; определяемое выражением

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n), \quad (4.27)$$

где m — общее число входов (за исключением порога) нейрона j; $w_{ji}(n)$ — синаптический вес, соединяющий нейроны i и j; $y_i(n)$ — входной сигнал нейрона j или выходной сигнал нейрона i. Если нейрон j расположен в первом скрытом слое сети, то $m = m_0$, а индекс i относится к i-му входному терминалу сети, для которого можно записать

$$y_i(n) = x_i(n), \tag{4.28}$$

где $x_i(n)$ — i-й элемент входного вектора (образа). С другой стороны, если нейрон j расположен в выходном слое сети, то $m = m_L$, а индекс j означает j-й выходной терминал сети, для которого можно записать

$$y_j(n) = o_j(n), \tag{4.29}$$

где $o_j(n)$ — j-й элемент выходного вектора. Выходной сигнал сравнивается с желаемым откликом $d_j(n)$, в результате чего вычисляется сигнал ошибки $e_j(n)$ для j-го выходного нейрона. Таким образом, прямая фаза вычислений начинается с представления входного вектора первому скрытому слою сети и завершается в последнем слое вычислением сигнала ошибки для каждого нейрона этого слоя.

Обратный проход начинается с выходного слоя предъявлением ему сигнала ошибки, который передается справа налево от слоя к слою с параллельным вычислением локального градиента для каждого нейрона. Этот рекурсивный процесс предполагает изменение синаптических весов в соответствии с дельта-правилом (4.25). Для нейрона, расположенного в выходном слое, локальный градиент равен соответствующему сигналу ошибки, умноженному на первую производную нелинейной функции активации. Затем соотношение (4.25) используется для вычисления изменений весов, связанных с выходным слоем нейронов. Зная локальные градиенты для всех нейронов выходного слоя, с помощью (4.24) можно вычислить локальные градиенты всех нейронов предыдущего слоя, а значит, и величину коррекции весов связей с этим слоем. Такие вычисления проводятся для всех слоев в обратном направлении.

Заметим, что после предъявления очередного обучающего примера он "фиксируется" на все время прямого и обратного проходов.

Функция активации

Вычисление локального градиента δ для каждого нейрона многослойного персептрона требует знания производной функции активации $\varphi(\cdot)$, связанной с этим нейроном. Для существования такой производной функция активации должна быть непрерывной. Другими словами, *дифференцируемость* является единственным требованием, которому должна удовлетворять функция активации. Примером непрерывно дифференцируемой нелинейной функции активации, которая часто используется в многослойных персептронах, является *сигмоидальная нелинейная функция* (sigmoidal nonlinearity), две формы которой описываются ниже.

1. Логистическая функция

Эта форма сигмоидальной нелинейности в общем виде определяется следующим образом:

$$\varphi_j(v_j(n)) = \frac{1}{1+\exp(-av_j(n))}, \quad a > 0 \ , \quad -\infty < v_j(n) < +\infty, \tag{4.30}$$

где $v_j(n)$ — индуцированное локальное поле нейрона j. Из (4.30) видно, что амплитуда выходного сигнала нейрона с такой активационной функцией лежит в диапазоне $0 \leq y_j \leq 1$. Дифференцируя (4.30) по $v_j(n)$, получим:

$$\varphi_j'(v_j(n)) = \frac{a\exp(-av_j(n))}{[1+\exp(-av_j(n))]^2}. \tag{4.31}$$

Так как $y_j(n) = \varphi_j(v_j(n))$, то можно избавиться от экспоненты $\exp(-av_j(n))$ в выражении (4.31) и представить производную функции активации в следующем виде:

$$\varphi_j'(v_j(n)) = ay_j(n)[1-y_j(n)]. \tag{4.32}$$

Для нейрона j, расположенного в выходном слое, $y_j(n) = o_j(n)$. Отсюда локальный градиент нейрона j можно выразить следующим образом:

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)) = a[d_j(n) - o_j(n)]o_j(n)[1-o_j(n)], \tag{4.33}$$

где $o_j(n)$ — функциональный сигнал на выходе нейрона j; $d_j(n)$ — его желаемый сигнал. С другой стороны, для произвольного скрытого нейрона j локальный градиент можно выразить так:

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n) = ay_j(n)[1-y_j(n)]\sum_k \delta_k(n)w_{kj}(n). \tag{4.34}$$

Обратите внимание, что, согласно выражению (4.32), производная $\varphi_j'(v_j(n))$ достигает своего максимального значения при $y_j(n)$ =0,5, а минимального (нуля) — при $y_j(n)$ =0 и $y_j(n)$ = 1,0. Так как величина коррекции синаптических весов в сети пропорциональна производной $\varphi_j'(v_j(n))$, то для максимального изменения синаптических весов нейрона его функциональные сигналы (вычисленные в соответствии с сигмоидальной функцией) должны находиться в середине диапазона. Согласно [914], именно это свойство алгоритма обратного распространения вносит наибольший вклад в его устойчивость как алгоритма обучения.

2. Функция гиперболического тангенса

Еще одной часто используемой формой сигмоидальной нелинейности является функция гиперболического тангенса, которая в общем виде описывается выражением

$$\varphi_j(v_j(n)) = a\tanh(bv_j(n)), \quad (a,\ b) > 0, \tag{4.35}$$

где a и b — константы. В действительности функция гиперболического тангенса является не более чем логистической функцией, только масштабированной и смещенной. Ее производная по аргументу $v_j(n)$ определяется следующим образом:

$$\begin{aligned}\varphi_j'(v_j(n)) &= ab\,\mathrm{sech}^2(bv_j(n)) = ab\left(1 - \tanh^2(bv_j(n))\right) = \\ &= \frac{b}{a}[a - y_j(n)][a + y_j(n)].\end{aligned} \tag{4.36}$$

Для нейрона j, расположенного в выходном слое, локальный градиент будет иметь вид

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)) = \frac{b}{a}[d_j(n) - o_j(n)][a - o_j(n)][a + o_j(n)]. \tag{4.37}$$

Для нейрона j, расположенного в скрытом слое, локальный градиент приобретает вид

$$\begin{aligned}\delta_j(n) &= \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n) = \\ &= \frac{b}{a}[a - y_j(n)][a + y_j(n)]\sum_k \delta_k(n)w_{kj}(n).\end{aligned} \tag{4.38}$$

Используя формулы (4.33) и (4.34) для логистической функции и формулы (4.37) и (4.38) для гиперболического тангенса, можно вычислить локальный градиент даже без явного знания активационной функции.

Скорость обучения

Алгоритм обратного распространения обеспечивает построение в пространстве весов "аппроксимации" для траектории, вычисляемой методом наискорейшего спуска. Чем меньше параметр скорости обучения η, тем меньше корректировка синаптических весов, осуществляемая на каждой итерации, и тем более гладкой является траектория в пространстве весов. Однако это улучшение происходит за счет замедления процесса обучения. С другой стороны, если увеличить параметр η для повышения скорости обучения, то результирующие большие изменения синаптических весов могут привести систему в неустойчивое состояние. Простейшим способом повышения скорости обучения без потери устойчивости является изменение дельта-правила (4.13) за счет

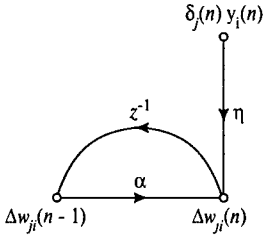


Рис. 4.6. Граф прохождения сигнала, иллюстрирующий эффект постоянной момента α

добавления к нему момента инерции[2] [914]:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n), \tag{4.39}$$

где α — как правило, положительное значение, называемое *постоянной момента* (momentum constant). Как показано на рис. 4.6, она является управляющим воздействием в контуре обратной связи для $\Delta w_{ji}(n)$, где z^{-1} — оператор единичной задержки. Уравнение (4.39) называется *обобщенным дельта-правилом*[3] (generalized delta rule). При α =0 оно вырождается в обычное дельта-правило.

Для того чтобы оценить влияние последовательности представления обучающих примеров на синаптические веса в зависимости от константы α, перепишем (4.39) в виде временного ряда с индексом t. Значение индекса t будет изменяться от нуля до текущего значения n. При этом выражение (4.39) можно рассматривать как разностное уравнение первого порядка для коррекции весов $\Delta w_{ji}(n)$. Решая это уравнение относительно $\Delta w_{ji}(n)$, получим временной ряд длины $n+1$:

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t). \tag{4.40}$$

Из уравнений (4.11) и (4.14) видно, что произведение $\delta_j(n) y_i(n) = -\partial \mathbf{E}(n)/\partial w_{ji}(n)$. Следовательно, соотношение (4.40) можно переписать в эквивалентной форме:

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial \mathbf{E}(t)}{\partial w_{ji}(t)}. \tag{4.41}$$

Основываясь на этом выражении, можно сделать следующие интуитивные наблюдения [505], [1117].

---

[2] Для частного случая алгоритма LMS было показано, что использование постоянной момента α уменьшает диапазон устойчивости параметра скорости обучения η и может привести к неустойчивости, если последний выбран неверно. Более того, с увеличением значения α рассогласование возрастает. Более подробно этот вопрос описан в [908].

[3] Вывод алгоритма обратного распространения с учетом постоянной момента содержится в [408].

1. Текущее значение коррекции весов $\Delta w_{ji}(n)$ представляет собой сумму экспоненциально взвешенного временного ряда. Для того чтобы этот ряд сходился, постоянная момента должна находиться в диапазоне $0 \leq |\alpha| < 1$. Если константа α равна нулю, алгоритм обратного распространения работает без момента. Следует также заметить, что константа α может быть и отрицательной, хотя эти значения не рекомендуется использовать на практике.
2. Если частная производная $\partial \mathbf{E}(n)/\partial w_{ji}(n)$ имеет один и тот же алгебраический знак на нескольких последовательных итерациях, то экспоненциально взвешенная сумма $\Delta w_{ji}(n)$ возрастает по абсолютному значению, поэтому веса $w_{ji}(n)$ могут изменяться на очень большую величину. Включение момента в алгоритм обратного распространения ведет к *ускорению спуска* (accelerate descent) в некотором постоянном направлении.
3. Если частная производная $\partial \mathbf{E}(n)/\partial w_{ji}(n)$ на нескольких последовательных итерациях меняет знак, экспоненциально взвешенная сумма $\Delta w_{ji}(n)$ уменьшается по абсолютной величине, поэтому веса $w_{ji}(n)$ изменяются на небольшую величину. Таким образом, добавление момента в алгоритм обратного распространения ведет к *стабилизирующему эффекту* (stabilized effect) для направлений, изменяющих знак.

Включение момента в алгоритм обратного распространения обеспечивает незначительную модификацию метода корректировки весов, оказывая положительное влияние на работу алгоритма обучения. Кроме того, слагаемое момента может предотвратить нежелательную остановку алгоритма в точке какого-либо локального минимума на поверхности ошибок.

При выводе алгоритма обратного распространения предполагалось, что параметр интенсивности обучения представлен константой η. Однако на практике он может задаваться как η_{ji}. Это значит, что параметр скорости обучения является локальным и определяется для каждой конкретной связи. В самом деле, применяя различные параметры скорости обучения в разных областях сети, можно добиться интересных результатов. Более подробно на этом вопросе мы остановимся в последующих разделах.

Следует отметить, что при реализации алгоритма обратного распространения можно изменять как все синаптические веса сети, так и только часть из них, оставляя остальные на время адаптации фиксированными. В последнем случае сигнал ошибки распространяется по сети в обычном порядке, однако фиксированные синаптические веса будут оставаться неизменными. Этого можно добиться, установив для соответствующих синаптических весов w_{ji} параметр интенсивности обучения η_{ji} равным нулю.

Последовательный и пакетный режимы обучения

В практических приложениях алгоритма обратного распространения в процессе обучения многослойного персептрона ему многократно предъявляется предопределенное множество обучающих примеров. Как уже отмечалось, один полный цикл предъявления полного набора примеров обучения называют *эпохой*. Процесс обучения проводится от эпохи к эпохе, пока синаптические веса и уровни порога не стабилизируются, а среднеквадратическая ошибка на всем обучающем множестве не сойдется к некоторому минимальному значению. Целесообразно *случайным образом изменять порядок представления примеров обучения* для разных эпох. Такой принцип предъявления образов делает поиск в пространстве весов стохастическим, предотвращая потенциальную возможность появления замкнутых циклов в процессе эволюции синаптических весов. Замкнутые циклы рассматриваются в главе 14.

Для данного обучающего множества алгоритм обратного распространения можно реализовать двумя способами.

1. Последовательный режим

Последовательный режим (sequential mode) обучения по методу обратного распространения также иногда называют *стохастическим* (stochastic) или *интерактивным* (on-line). В этом режиме корректировка весов проводится после подачи каждого примера. Это тот самый режим, для которого мы выводили алгоритм обратного распространения ранее в этой главе. Для примера рассмотрим эпоху, состоящую из N обучающих примеров, упорядоченных следующим образом: $(\mathbf{x}(1), \mathbf{d}(1)), \ldots, (\mathbf{x}(N), \mathbf{d}(N))$. Сети предъявляется первый пример $(\mathbf{x}(1), \mathbf{d}(1))$ этой эпохи, после чего выполняются описанные выше прямые и обратные вычисления. В результате проводится корректировка синаптических весов и уровней порогов в сети. После этого сети предъявляется вторая пара $(\mathbf{x}(2), \mathbf{d}(2))$ в эпохе, повторяются прямой и обратный проходы, приводящие к следующей коррекции синаптических весов и уровня порога. Этот процесс повторяется, пока сеть не завершит обработку последнего примера (пары) данной эпохи – $(\mathbf{x}(N), \mathbf{d}(N))$.

2. Пакетный режим

В *пакетном режиме* (batch mode) обучения по методу обратного распространения корректировка весов проводится после подачи в сеть примеров обучения (эпохи). Для конкретной эпохи функция стоимости определяется как среднеквадратическая ошибка (4.2) и (4.3), представленная в составной форме:

$$\mathrm{E}_{\mathrm{av}}(n) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n), \tag{4.42}$$

где сигнал ошибки $e_j(n)$ соответствует нейрону j для примера обучения n и определяется формулой (4.1). Ошибка $e_j(n)$ равна разности между $d_j(n)$ и $y_j(n)$ для j-го элемента вектора желаемых откликов $\mathbf{d}(n)$ и соответствующего выходного нейрона сети. В выражении (4.42) внутреннее суммирование по j выполняется по всем нейронам выходного слоя сети, в то время как внешнее суммирование по n выполняется по всем образам данной эпохи. При заданном параметре скорости обучения η корректировка, применяемая к синаптическому весу w_{ji}, связывающему нейроны i и j, определяется следующим дельта-правилом:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathbf{E}_{\text{av}}}{\partial w_{ji}} = -\frac{\eta}{N} \sum_{n=1}^{N} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}}. \tag{4.43}$$

Для вычисления частной производной $\partial e_j(n)/\partial w_{ji}$ нужно проделать тот же путь, что и ранее. Согласно (4.43), в пакетном режиме корректировка веса Δw_{ji} выполняется только после прохождения по сети всего множества примеров.

С точки зрения процессов реального времени, последовательный режим является более предпочтительным, чем пакетный, так как требует *меньшего* объема внутреннего хранилища для каждой синаптической связи. Более того, предъявляя обучающие примеры в случайном порядке (в процессе последовательной корректировки весов), поиск в пространстве весов можно сделать действительно *стохастическим*. Это, в свою очередь, сокращает до минимума возможность остановки алгоритма в точке какого-либо локального минимума.

Следует отметить, что стохастическая природа последовательного режима усложняет построение теоретического фундамента для нахождения условий сходимости алгоритма. В противовес этому использование пакетного режима обеспечивает точную оценку вектора градиента. Таким образом, сходимость алгоритма к локальному минимуму гарантируется при довольно простых условиях. Помимо того, в пакетном режиме легче распараллелить вычисления.

Если данные обучения являются *избыточными* (redundant) (т.е. содержат по несколько копий одних и тех же примеров), то предпочтительнее использовать последовательный режим, так как примеры все равно подаются по одному. Это преимущество особенно заметно при больших наборах данных с высокой степенью избыточности.

В заключение можно сказать, что несмотря на многие недостатки последовательного режима алгоритма обратного распространения, он остается очень популярным (особенно при решении задач распознавания образов) по двум практическим причинам.

- Этот алгоритм прост в реализации.
- Обеспечивает эффективное решение сложных и больших задач.

Критерий останова

В общем случае не существует доказательства сходимости алгоритма обратного распространения, как не существует и какого-либо четко определенного критерия его останова. Известно лишь несколько обоснованных критериев, которые можно использовать для прекращения корректировки весов. Каждый из них имеет свои практические преимущества. Для того чтобы сформулировать такой критерий, вполне логично рассуждать в терминах уникальных свойств *локального* (local) и *глобального* (global) минимумов поверхности ошибок[4]. Обозначим символом $\mathbf{w}^*$ вектор весов, обеспечивающий минимум, будь то локальный или глобальный. Необходимым условием минимума является то, что вектор градиента $\mathbf{g}(\mathbf{w})$ (т.е. вектор частных производных первого порядка) для поверхности ошибок в этой точке равен нулевому. Следовательно, можно сформулировать разумный критерий сходимости алгоритма обучения обратного распространения [596].

Считается, что алгоритм обратного распространения сошелся, если Евклидова норма вектора градиента достигает достаточно малых значений.

Недостатком этого критерия сходимости является то, что для сходимости обучения может потребоваться довольно много времени. Кроме того, необходимо постоянно вычислять вектор градиента $\mathbf{g}(\mathbf{w})$.

Другим уникальным свойством минимума является то, что функция стоимости (или мера ошибки) $\mathbf{E}_{\text{av}}(\mathbf{w})$ в точке $\mathbf{w} = \mathbf{w}^*$ стабилизируется. Отсюда можно вывести еще один критерий сходимости.

Критерием сходимости алгоритма обратного распространения является достаточно малая абсолютная интенсивность изменений среднеквадратической ошибки в течение эпохи.

Интенсивность изменения среднеквадратической ошибки обычно считается достаточно малой, если она лежит в пределах от 0,1–1% за эпоху. Иногда используется уменьшенное значение — 0,01%. К сожалению, этот критерий может привести к преждевременной остановке процесса обучения.

Существует еще один полезный и теоретически подкрепленный критерий сходимости. После каждой итерации обучения сеть тестируется на эффективность обобщения. Процесс обучения останавливается, когда эффективность обобщения становится удовлетворительной или когда оказывается, что пик эффективности уже пройден. В разделе 4.14 мы поговорим об этом более подробно.

---

[4] Вектор $\mathbf{w}^*$ называется *локальным минимумом* функции F, если в этой точке она достигает наименьшего значения в некоторой области, т.е. если существует такое значениеε, что [125] $F(\mathbf{w}^*) \leq F(\mathbf{w})$ для всех $\mathbf{w}$, удовлетворяющих неравенству $||\mathbf{w} - \mathbf{w}^*|| < \varepsilon$. Вектор $\mathbf{w}^*$ иазывается *глобальным минимумом* функции F, если в этой точке она достигает наименьшего значения на всей своей области определения, т.е. если $F(\mathbf{w}^*) \leq F(\mathbf{w})$ для всех $\mathbf{w} \in \Re^n$, где n — размериость вектора $\mathbf{w}$.

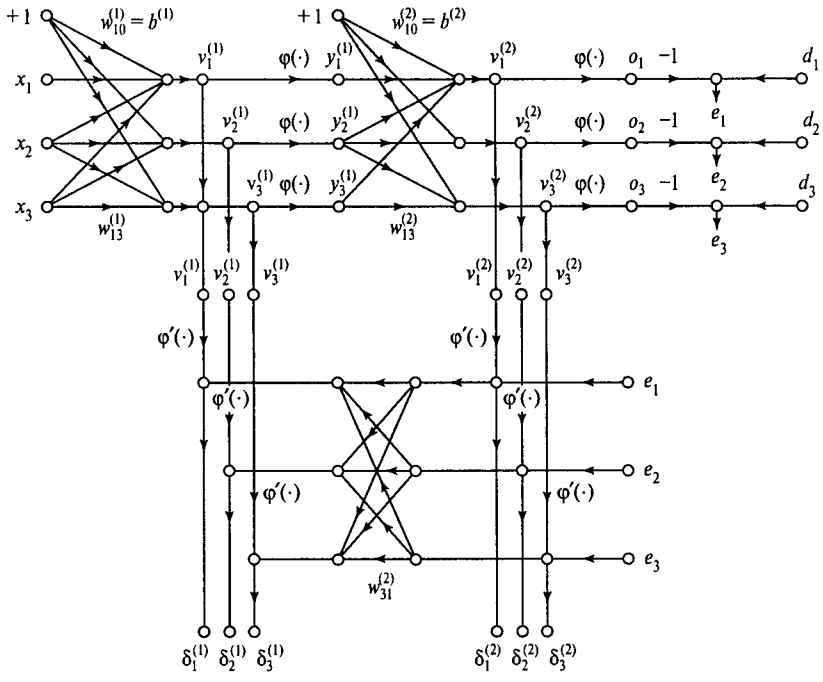


Рис. 4.7. Граф передачи сигнала для процесса обучения по методу обратного распространения. Верхняя часть графа — прямой проход, нижняя часть графа — обратный проход

4.4. Алгоритм обратного распространения в краткой форме

На рис. 4.1 представлена архитектурная схема многослойного персептрона. Соответствующий граф передачи сигнала в процессе обучения по методу обратного распространения, иллюстрирующий как прямую, так и обратную фазу вычислений, представлен на рис. 4.7 для случая $L = 2$ и $m_0 = m_1 = m_2 = 3$. В верхней части графа передачи сигнала показан прямой проход, в нижней — обратный. Последний еще носит название *графа чувствительности* (sensitivity graph) для вычисления локальных градиентов в алгоритме обратного распространения [774].

Ранее мы упоминали, что последовательная корректировка весов является более предпочтительным режимом алгоритма обратного распространения для реализации в реальном времени. В этом режиме алгоритм циклически обрабатывает примеры из обучающего множества $\{(x(n), d(n))\}_{n=1}^{N}$ следующим образом.

1. *Инициализация* (initialization). Предполагая отсутствие априорной информации, генерируем синаптические веса и пороговые значения с помощью датчика рав-

номерно распределенных чисел со средним значением 0. Дисперсия выбирается таким образом, чтобы стандартное отклонение индуцированного локального поля нейронов приходилось на линейную часть сигмоидальной функции активации (и не достигало области насыщения).

2. *Предъявление примеров обучения* (presentation of training examples). В сеть подаются образы из обучающего множества (эпохи). Для каждого образа последовательно выполняются прямой и обратный проходы, описанные далее в пп. 3 и 4.

3. *Прямой проход* (forward computation). Пусть пример обучения представлен парой $(\mathbf{x}(n), \mathbf{d}(n))$, где $\mathbf{x}(n)$ — входной вектор, предъявляемый входному слою сенсорных узлов; $\mathbf{d}(n)$ — желаемый отклик, предоставляемый выходному слою нейронов для формирования сигнала ошибки. Вычисляем индуцированные локальные поля и функциональные сигналы сети, проходя по ней послойно в прямом направлении. Индуцированное локальное поле нейрона j слоя l вычисляется по формуле

$$v_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n) y_i^{(l-1)}(n), \tag{4.44}$$

где $y_i^{(l-1)}(n)$ — выходной (функциональный) сигнал нейрона i, расположенного в предыдущем слое $l-1$, на итерации n; $w_{ji}^{(l)}(n)$ —синаптический вес связи нейрона j слоя l с нейроном i слоя $l-1$. Для $i = 0$ $y_0^{(l-1)}(n) = +1$, а $w_{j0}^{(l)}(n) = b_j^l(n)$ — порог, применяемый к нейрону j слоя l. Если используется сигмоидальная функция, то выходной сигнал нейрона j слоя l выражается следующим образом:

$$y_j^{(l)}(n) = \varphi_j(v_j(n)).$$

Если нейрон j находится в первом скрытом слое (т.е. $l = 1$), то

$$y_j^{(0)}(n) = x_j(n),$$

где $x_j(n)$ — j-й элемент входного вектора $\mathbf{x}(n)$. Если нейрон j находится в выходном слое (т.е. $l = L$, где L — глубина сети), то

$$y_j^{(L)}(n) = o_j(n).$$

Вычисляем сигнал ошибки

$$e_j(n) = d_j(n) - o_j(n), \tag{4.45}$$

где $d_j(n)$ — j-й элемент вектора желаемого отклика $\mathbf{d}(n)$.

4. *Обратный проход* (backward computation). Вычисляем локальные градиенты узлов сети по следующей формуле:

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\varphi_j'(v_j^{(L)}(n)) & \text{для нейрона } j \text{ выходного слоя } L, \\ \varphi_j'(v_j^{(l)}(n)) \sum\limits_k \delta_k^{(l+1)}(n) \delta_{kj}^{(l+1)}(n) & \text{для нейрона } j \text{ скрытого слоя } l, \end{cases} \quad (4.46)$$

где штрих в функции $\varphi_j'(\cdot)$ обозначает дифференцирование по аргументу. Изменение синаптических весов слоя l сети выполняется в соответствии с обобщенным дельта-правилом

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[w_{ji}^{(l)}(n-1)] + \eta\delta_j^{(l)}(n) y_j^{(l-1)}(n), \quad (4.47)$$

где η — параметр скорости обучения; α — постоянная момента.

5. *Итерации* (iteration). Последовательно выполняем прямой и обратный проходы (согласно пп. 3, 4), предъявляя сети все примеры обучения из эпохи, пока не будет достигнут критерий останова.

Примечание. Порядок представления примеров обучения может случайным образом меняться от эпохи к эпохе. Параметры момента и скорости обучения настраиваются (и обычно уменьшаются) по мере роста количества итераций. Объяснение этого факта будет приведено ниже.

4.5. Задача XOR

В элементарном (однослойном) персептроне нет скрытых нейронов. Следовательно, он не может классифицировать входные образы, которые линейно-неразделимы. Однако нелинейно-разделимые области встречаются не так уж редко. Например, именно такая проблема возникает в известной *задаче "исключающего ИЛИ"* (XOR — exclusive OR problem), которую можно рассматривать как частный случай более общей *задачи классификации точек единичного гиперкуба* (unit hypercube). Каждая точка этого куба принадлежит одному из двух классов — 0 или 1. Для частного случая задачи XOR рассмотрим четыре угла *единичного квадрата*, которые соответствуют входным парам $(0,0)$, $(0,1)$, $(1,0)$ и $(1,1)$. Первая и четвертая пары принадлежат классу 0, т.е.

$$0 \oplus 0 = 0 \text{ и } 1 \oplus 1 = 0,$$

где знак $\oplus$ обозначает оператор булевой функции XOR. Входные образы $(0,0)$ и $(1,1)$ располагаются в противоположных углах *единичного квадрата* (unit square), но генерируют одинаковый выходной сигнал, равный нулю. С другой стороны, пары $(0,1)$ и $(1,0)$ также располагаются в противоположных углах квадрата, но принадлежат классу 1:

$$0 \oplus 1 = 1 \text{ и } 1 \oplus 0 = 1.$$

Очевидно, что при использовании всего одного нейрона с двумя входами граница решений в пространстве входов будет линейной. Для всех точек, расположенных по одну сторону этой линии, выход нейрона будет равен единице, а для остальных точек — нулю. Положение и ориентация этой линии в пространстве входов определяется синаптическими весами, соединяющими нейрон с входными узлами, и порогом. Имея две пары точек, расположенных в противоположных углах квадрата и относящихся к разным классам, мы, естественно, не сможем построить прямую линию так, чтобы точки одного класса лежали по одну сторону от разделяющей поверхности. Другими словами, элементарный персептрон не может решить задачу XOR.

Для решения этой задачи введем скрытый слой с двумя нейронами (рис. 4.8, *а*) [1058]. Граф передачи сигнала этой сети показан на рис. 4.8, *б*. При этом сделаем следующие допущения.

- Каждый из нейронов представлен моделью Мак-Каллока–Питца, в которой функцией активации является пороговая.
- Биты 0 и 1 представлены соответственно уровнями сигнала 0 и +1.

Верхний нейрон (с меткой 1 в скрытом слое) характеризуется следующим образом:

$$w_{11} = w_{12} = +1, b_1 = -3/2.$$

Наклон границы решений для этого скрытого нейрона равен -1, что и показано на рис. 4.9, *а*. Нижний нейрон (с меткой 2 в скрытом слое) обладает свойствами

$$w_{21} = w_{22} = +1, b_2 = -1/2.$$

Ориентация и расположение границы решений для второго скрытого нейрона показаны на рис. 4.9, *б*.

Для выходного нейрона (обозначенного меткой 3 на рис. 4.8, *а*)

$$w_{31} = -2, w_{32} = +1, b_1 = -1/2.$$

Функцией выходного нейрона является построение линейной комбинации границ решений, сформированных двумя скрытыми нейронами. Результат вычислений показан на рис. 4.9, *в*. Нижний скрытый нейрон соединен возбуждающей (положительной) связью с выходным нейроном, в то время как верхний скрытый нейрон — тормозящей (отрицательной). Если оба скрытых нейрона заторможены (что соответствует вход-

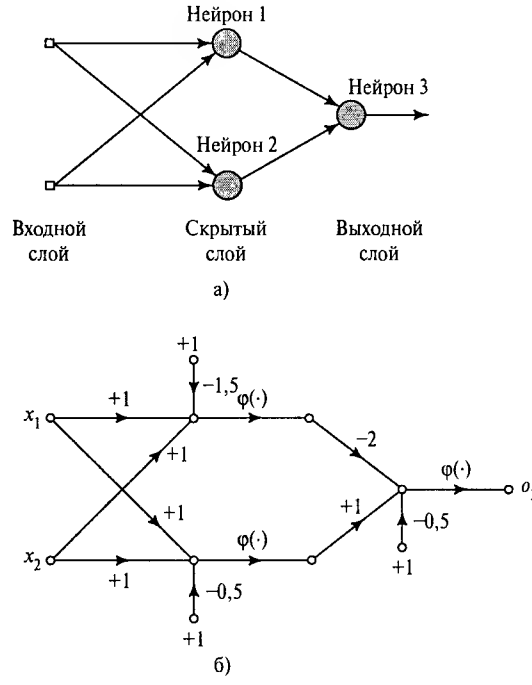


Рис. 4.8. Архитектурный граф сети для решения задачи XOR (а) и граф передачи сигнала для этой сети (б) .

ному образу (0, 0)), выходной нейрон также остается неактивным. Если оба скрытых нейрона возбуждены (что соответствует входному образу (1, 1)), выходной нейрон остается неактивным, так как тормозящее влияние большого отрицательного веса (верхний скрытый нейрон) преобладает над возбуждающим воздействием с меньшим весом (нижний скрытый нейрон). Когда верхний скрытый нейрон находится в заторможенном состоянии, а нижний — в возбужденном (что соответствует входным образам (0, 1) и (1, 0)), выходной нейрон переходит в возбужденное состояние, так как возбуждающий сигнал с положительным весом приходит от нижнего скрытого нейрона. Таким образом, сеть, изображенная на рис. 4.8, *а*, и в самом деле решает задачу XOR.

4.6. Эвристические рекомендации по улучшению работы алгоритма обратного распространения

Часто утверждают, что проектирование нейронных сетей, использующих алгоритм обратного распространения, является скорее искусством, чем наукой. При этом имеют в виду тот факт, что многочисленные параметры этого процесса определяются только на основе личного практического опыта разработчика. В этом утверждении есть доля правды. Тем не менее приведем некоторые общие методы, улучшающие производительность алгоритма обратного распространения.

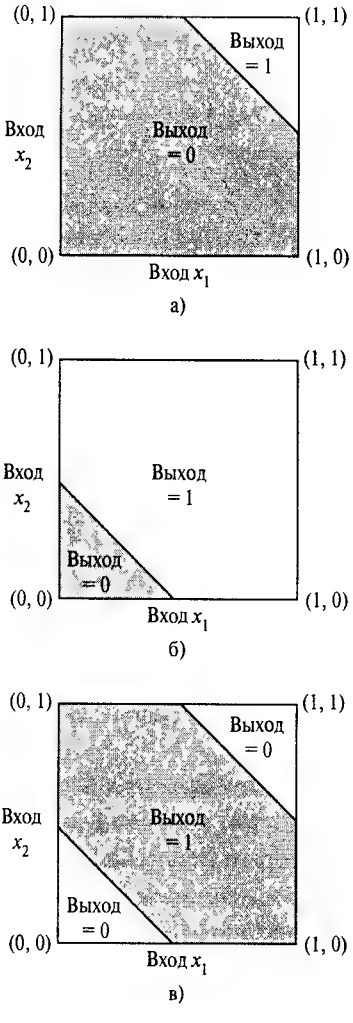


Рис. 4.9. Границы решений, построенные для скрытого нейрона 1 сети, показанной на рис. 4.8, *а* (а); границы решений для скрытого нейрона 2 сети (б) и для всей сети (в)

1. *Режим: последовательный или пакетный* (sequential versus batch update). Как уже говорилось ранее, последовательный режим обучения методом обратного распространения (использующий последовательное предоставление примеров эпохи с обновлением весов на каждом шаге) в вычислительном смысле оказывается значительно быстрее. Это особенно сказывается тогда, когда обучающее множество является большим и в высокой степени избыточным. (Избыточные данные вызывают вычислительные проблемы при оценке Якобиана, необходимой для пакетного режима.)
2. *Максимизация информативности* (maximizing information content). Как правило, каждый обучающий пример, предоставляемый алгоритму обратного распространения, нужно выбирать из соображений наибольшей информационной насыщенности в области решаемой задачи [617]. Для этого существуют два общих метода.

- Использование примеров, вызывающих наибольшие ошибки обучения.
- Использование примеров, которые радикально отличаются от ранее использованных.

Эти два эвристических правила мотивированы желанием максимально расширить область поиска в пространстве весов.

В задачах классификации, основанных на последовательном обучении методом обратного распространения, обычно применяется метод случайного изменения порядка следования примеров, подаваемых на вход многослойного персептрона, от одной эпохи к другой. В идеале такая рандомизация приводит к тому, что успешно обрабатываемые примеры будут принадлежать к различным классам.

Более утонченным приемом является *схема акцентирования* (emphasizing scheme), согласно которой более сложные примеры подаются в систему чаще, чем более легкие [617]. Простота или сложность отдельных примеров выявляется с помощью анализа динамики ошибок (в разрезе итераций), генерируемых системой при обработке обучающих примеров. Однако использование схемы акцентирования приводит к двум проблемам, которые следует учесть.

- Распределение примеров в эпохе, представляемой сети, искажается.
- Наличие исключений или немаркированных примеров может привести к катастрофическим последствиям с точки зрения эффективности алгоритма. Обучение на таких исключениях подвергает риску способность сети к обобщению в наиболее правдоподобных областях пространства входных сигналов.

3. *Функция активации* (activation function). Многослойный персептрон, обучаемый по алгоритму обратного распространения, может в принципе обучаться быстрее (в терминах требуемого для обучения количества итераций), если сигмоидальная функция активации нейронов сети является антисимметричной, а не симметричной. Более подробно этот вопрос рассматривается в разделе 4.11. Функция активации $\varphi(v)$ называется *антисимметричной* (т.е. четной функцией своего аргумента), если

$$\varphi(-v) = -\varphi(v),$$

что показано на рис. 4.10, *а*. Стандартная логистическая функция не удовлетворяет этому условию (рис. 4.10, *б*).

Известным примером антисимметричной функции активации является сигмоидальная нелинейная функция *гиперболического тангенса* (hyperbolic tangent)

$$\varphi(v) = a \tanh(bv),$$

где a и b — константы. Удобными значениями для констант a и b являются следующие [617], [618]:

$$a = 1,7159,$$
$$b = 2/3.$$

Определенная таким образом функция гиперболического тангенса имеет ряд полезных свойств.

- $\varphi(1) = 1$ и $\varphi(-1) = -1$.
- В начале координат тангенс угла наклона (т.е. эффективный угол) функции активации близок к единице:

$$\varphi(0) = ab = 1,7159 \times 2/3 = 1,1424.$$

- Вторая производная $\varphi(v)$ достигает своего максимального значения при $v = 1$.

4. *Целевые значения* (target value). Очень важно, чтобы целевые значения выбирались из области значений сигмоидальной функции активации. Более точно, желаемый отклик d_j нейрона j выходного слоя многослойного персептрона должен быть *смещен* на некоторую величину ε от границы области значений функции активации в сторону ее внутренней части. В противном случае алгоритм обратного распространения будет модифицировать свободные параметры сети, устремляя их в бесконечность, замедляя таким образом процесс обучения и доводя скрытые нейроны до предела насыщения. В качестве примера рассмотрим антисимметричную функцию активации, показанную на рис. 4.10, *а*. Для предельного значения $+a$ выберем

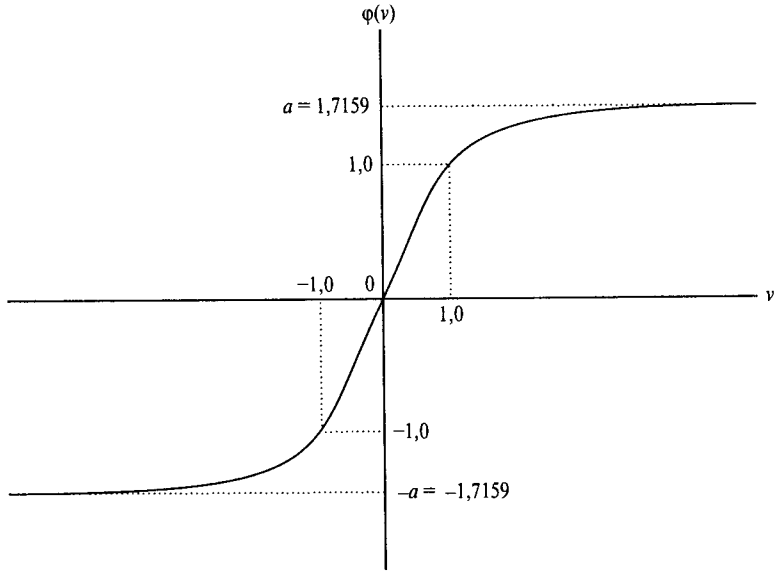
$$d_j = a - \varepsilon.$$

Аналогично, для предельного значения $-a$ установим

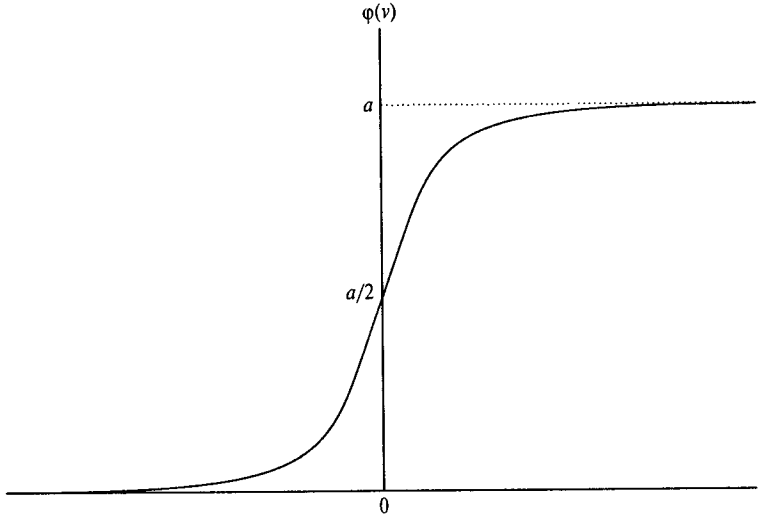
$$d_j = -a + \varepsilon,$$

где ε — соответствующая положительная константа. Для выбранного ранее значения $a = 1,7159$ установим $\varepsilon = 0,7159$. В этом случае желаемый отклик d_j будет находиться в диапазоне от -1 до $+1$ (см. рис. 4.10, *а*).

5. *Нормализация входов* (normalizing the inputs). Все входные переменные должны быть *предварительно обработаны* так, чтобы среднее значение по всему обучающему множеству было близко к нулю, иначе их будет сложно сравнивать со стандартным отклонением [617]. Для оценки практической значимости этого правила рассмотрим экстремальный случай, когда все входные переменные положительны. В этом случае синаптические веса нейрона первого скрытого слоя могут либо одновременно увеличиваться, либо одновременно уменьшаться. Следовательно, вектор весов этого нейрона будет менять направление, что приведет



а)



б)

Рис. 4.10. Антисимметричная (а) и асимметричная (б) функции активации

к зигзагообразному движению по поверхности ошибки. Такая ситуация обычно замедляет процесс обучения и, таким образом, неприемлема.

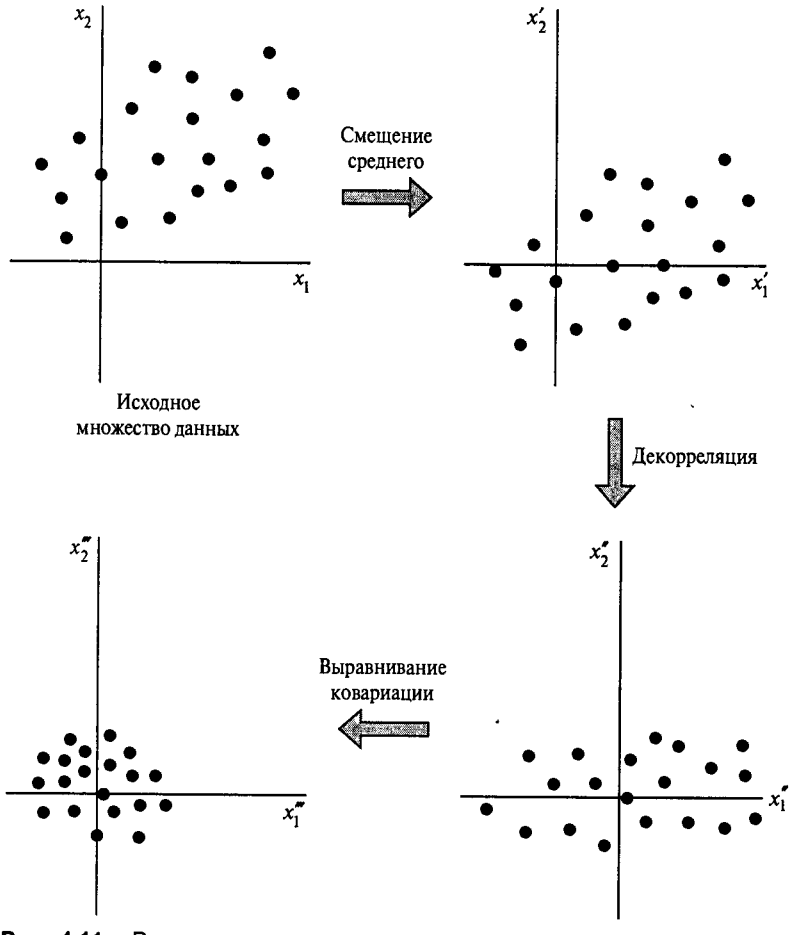


Рис. 4.11. Результаты трех шагов нормализации: смещения среднего, декорреляции и выравнивания ковариации

Чтобы ускорить процесс обучения методом обратного распространения, входные векторы необходимо нормализовать в двух следующих аспектах [617].

- Входные переменные, содержащиеся в обучающем множестве, должны быть *некоррелированны* (uncorrelated). Этого можно добиться с помощью анализа главных компонентов, который детально описывается в главе 8.
- Некоррелированные входные переменные должны быть масштабированы так, чтобы их *ковариация была приближенно равной* (approximately equal). Тогда различные синаптические веса сети будут обучаться приблизительно с одной скоростью.

На рис. 4.11 показан результат трех шагов нормализации: смещения среднего, декорреляции и выравнивания ковариации, примененных в указанном порядке.

6. *Инициализация* (initialization). Хороший выбор начальных значений синаптических весов и пороговых значений (threshold) сети может оказать неоценимую помощь в проектировании. Естественно, возникает вопрос: "А что такое хорошо?"

   Если синаптические веса принимают большие начальные значения, то нейроны, скорее всего, достигнут режима насыщения. Если такое случится, то локальные градиенты алгоритма обратного распространения будут принимать малые значения, что, в свою очередь, вызовет торможение процесса обучения. Если же синаптическим весам присвоить малые начальные значения, алгоритм будет очень вяло работать в окрестности начала координат поверхности ошибок. В частности, это верно для случая антисимметричной функции активации, такой как гиперболический тангенс. К сожалению, начало координат является *седловой точкой* (saddle point), т.е. стационарной точкой, где образующие поверхности ошибок вдоль одной оси имеют положительный градиент, а вдоль другой — отрицательный. По этим причинам нет смысла использовать как слишком большие, так и слишком маленькие начальные значения синаптических весов. Как всегда, золотая середина находится между этими крайностями.

   Для примера рассмотрим многослойный персептрон, в котором в качестве функции активации используется гиперболический тангенс. Пусть пороговое значение, применяемое к нейронам сети, равно нулю. Исходя из этого, индуцированное локальное поле нейрона j можно выразить следующим образом:

$$v_j = \sum_{i=1}^{m} w_{ji} y_i.$$

   Предположим, что входные значения, передаваемые нейронам сети, имеют нулевое среднее значение и дисперсию, равную единице, т.е.

$$\mu_y = E[y_i] = 0 \text{ для всех } i,$$
$$\sigma_y^2 = E[(y_i - \mu_i)^2] = E[y_i^2] = 1 \text{ для всех } i.$$

   Далее предположим, что входные сигналы некоррелированны:

$$E[y_i y_k] = \begin{cases} 1 \text{ для } k = i, \\ 0 \text{ для } k \neq i, \end{cases}$$

   и синаптические веса выбраны из множества равномерно распределенных чисел с нулевым средним

$$\mu_w = E[w_{ji}] = 0 \text{ для всех пар } (j, i)$$

и дисперсией

$$\sigma_w^2 = E[(w_{ji} - \mu_w)^2] = E[w_{ji}^2] \text{ для всех пар } (j, i).$$

Следовательно, математическое ожидание и дисперсию индуцированного локального поля можно выразить так:

$$\begin{aligned}
\mu_v &= E[v_j] = E\left[\sum_{i=1}^{m} w_{ji} y_i\right] = \sum_{i=1}^{m} E[w_{ji}] E[y_i] = 0, \\
\sigma_v^2 &= E[(v_j - \mu_v)^2] = E[v_j^2] = E\left[\sum_{i=1}^{m}\sum_{k=1}^{m} w_{ji} w_{jk} y_i y_k\right] = \\
&= \sum_{i=1}^{m}\sum_{k=1}^{m} E[w_{ji} w_{jk}] E[y_i y_k] = \sum_{i=1}^{m} E[w_{ji}^2] = m\sigma_w^2,
\end{aligned} \tag{4.48}$$

где m — число синаптических связей нейрона.

На основании этого результата можно описать хорошую стратегию инициализации синаптических весов таким образом, чтобы стандартное отклонение индуцированного локального поля нейрона лежало в переходной области между линейной частью сигмоидальной функции активации и областью насыщения. Например, для случая гиперболического тангенса с параметрами a и b (см. определение функции) эта цель достигается при $\sigma_v = 1$ в (4.48). Исходя из этого, получим [617]:

$$\sigma_w = m^{-1/2}. \tag{4.49}$$

Таким образом, желательно, чтобы равномерное распределение, из которого выбираются исходные значения синаптических весов, имело нулевое среднее значение и дисперсию, обратную корню квадратному из количества синаптических связей нейрона.

7. *Обучение по подсказке* (hints). Обучение на множестве примеров связано с аппроксимацией неизвестной функцией отображения входного сигнала на выходной. В процессе обучения из примеров извлекается информация о функции $f(\cdot)$ и строится некоторая аппроксимация этой функциональной зависимости. Процесс обучения на примерах можно обобщить, добавив *обучение по подсказке*, которое реализуется путем предоставления некоторой априорной информации о функции $f(\cdot)$ [4]. Такая информация может включать свойства инвариантности, симметрии и прочие знания о функции $f(\cdot)$, которые можно использовать для ускорения поиска ее аппроксимации и, что более важно, для повышения качества конечной оценки. Использование соотношения (4.49) является одним из примеров такого подхода.

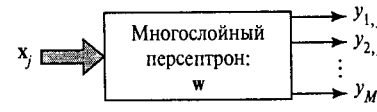


Рис. 4.12. Блочная диаграмма классификатора входных сигналов

8. *Скорость обучения* (learning rates). Все нейроны многослойного персептрона в идеале должны обучаться с одинаковой скоростью. Однако последние слои обычно имеют более высокие значения локальных градиентов, чем начальные слои сети. Исходя из этого параметру скорости обучения η следует назначать меньшие значения для последних слоев сети и большие — для первых. Чтобы время обучения для всех нейронов сети было примерно одинаковым, нейроны с большим числом входов должны иметь меньшее значение параметра обучения, чем нейроны с малым количеством входов. В [617] предлагается назначать параметр скорости обучения для каждого нейрона обратно пропорционально квадратному корню из суммы его синаптических связей. Более подробно о параметре скорости обучения речь пойдет в разделе 4.17.

4.7. Представление выхода и решающее правило

Теоретически для *задачи классификации на M классов* (M-class classification problem), в которой объединение M классов формирует все пространство входных сигналов, для представления всех возможных результатов классификации требуется M выходов (рис. 4.12). На этом рисунке вектор $\mathbf{x}_j$ является j-м *прототипом* (prototype) (т.е. отдельной реализацией) m-мерного случайного вектора $\mathbf{x}$, который должен быть классифицирован многослойным персептроном. k-й из M возможных классов, которому принадлежит данный входной сигнал, обозначается $\mathbf{C}_k$. Пусть y_{kj} — k-й выход сети, генерируемый в ответ на прототип $\mathbf{x}_j$:

$$y_{k,j} = F_k(\mathbf{x}_j), k = 1, 2, \ldots, M, \tag{4.50}$$

где функция $F_k(\cdot)$ определяет отображение, которому обучается сеть при передаче входного примера на k-й выход. Для удобства представления обозначим

$$\mathbf{y}_j = [y_{1,j}, y_{2,j}, \ldots, y_{M,j}]^T = [F_1(\mathbf{x}_j), F_2(\mathbf{x}_j), \ldots, F_M(\mathbf{x}_j)]^T = \mathbf{F}(\mathbf{x}_j), \tag{4.51}$$

где $\mathbf{F}(\cdot)$ — вектор-функция.

Главный вопрос этого раздела звучит так.

Каким должно быть оптимальное решающее правило, применяемое для классификации M выходов сети после обучения многослойного персептрона?

Естественно, решающее правило должно основываться на знании вектор-функции

$$\mathbf{F} : \Re^m \mathbf{x} \rightarrow \mathbf{y} \in \Re^M. \tag{4.52}$$

В общем случае о вектор-функции $\mathbf{F}(\cdot)$ определенно известно лишь то, что это непрерывная функция, минимизирующая *функционал эмпирического риска* (empirical risk functional):

$$R = \frac{1}{2N} \sum_{j=1}^{N} \|\mathbf{d}_j - \mathbf{F}(\mathbf{x}_j)\|^2, \tag{4.53}$$

где $\mathbf{d}_j$ — желаемый (целевой) выход для прототипа $\mathbf{x}_j$; $||\cdot||$ — Евклидова норма вектора; N — общее число примеров, представленных сети для обучения. Сущность критерия (4.53) та же, что и у функции стоимости (4.3). Вектор-функция $\mathbf{F}(\cdot)$ строго зависит от выбора примеров $(\mathbf{x}_j, \mathbf{d}_j)$, использованных для обучения сети. Это значит, что разные значения пар $(\mathbf{x}_j, \mathbf{d}_j)$ приведут к построению различных вектор-функций $\mathbf{F}(\cdot)$. Обратите внимание, что используемое здесь обозначение $(\mathbf{x}_j, \mathbf{d}_j)$ является эквивалентом употреблявшегося ранее обозначения $(\mathbf{x}(j), \mathbf{d}(j))$.

Предположим, что сеть обучается на двоичных целевых значениях (которые случайно совпадают с верхней и нижней границами области значений логистической функции):

$$d_{kj} = \begin{cases} 1, & \text{если прототип } \mathbf{x}_j \text{ принадлежит классу } \mathbf{C}_k, \\ 0, & \text{если прототип } \mathbf{x}_j \text{ не принадлежит классу } \mathbf{C}_k. \end{cases} \tag{4.54}$$

Основываясь на этом допущении, класс $\mathbf{C}_k$ можно представить M-мерным целевым вектором

$$\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow k\text{-й элемент.}$$

Напрашивается предположение, что многослойный классификатор персептронного типа, обученный по алгоритму обратного распространения на конечном множестве независимых и равномерно распределенных примеров, обеспечивает асимптотическую аппроксимацию соответствующей апостериорной вероятности класса. Это свойство можно обосновать следующим образом [881], [1133].

- Согласно закону больших чисел, при бесконечном увеличении размера N обучающего множества вектор $\mathbf{w}$, минимизирующий функционал стоимости R из (4.53), достигает оптимального значения $\mathbf{w}^*$, минимизирующего ожидание случайной величины $1/2||\mathbf{d} - \mathbf{F}(\mathbf{w}, \mathbf{x})||^2$, где $\mathbf{d}$ — вектор желаемого отклика; $\mathbf{F}(\mathbf{w}, \mathbf{x})$ — аппроксимация, реализованная многослойным персептроном для вектора весовых коэффициентов $\mathbf{w}$ и входа $\mathbf{x}$ [1133]. Функция $\mathbf{F}(\mathbf{w}, \mathbf{x})$, в которой явным образом показана зависимость от вектора $\mathbf{w}$, — это не что иное, как использованная ранее функция $\mathbf{F}(\mathbf{x})$.
- Оптимальный вектор весов $\mathbf{w}^*$ обладает тем свойством, что соответствующий ему вектор фактического выхода сети $\mathbf{F}(\mathbf{w}^*, \mathbf{x})$ является аппроксимацией, построенной по методу наименьших квадратов и минимизирующей ошибку условного ожидания вектора желаемого отклика при данном входном векторе $\mathbf{x}$ [1133]. Этот вопрос уже обсуждался в главе 2.
- Для задачи классификации входных сигналов на M классов k-й элемент вектора желаемого отклика равен единице, если входной вектор $\mathbf{x}$ принадлежит к классу $\mathbf{C}_k$, и нулю в противном случае. Отсюда следует, что условное ожидание вектора желаемого отклика при данном векторе $\boldsymbol{x}$ равно апостериорной вероятности класса $P(\mathbf{C}_k|\mathbf{x})$, $k = 1, 2, \ldots, M$ [881].

Отсюда следует, что многослойный персептрон (с логистической активационной функцией) действительно аппроксимирует *апостериорную* (a posteriori) вероятность распознавания класса при условии, что размерность обучающего множества достаточно велика и что процесс обучения методом обратного распространения не прекратится в точке локального минимума. Теперь можно ответить на поставленный ранее вопрос. В частности, можно утверждать, что соответствующее решающее правило является (приближенно) байесовским правилом, обобщенным для *апостериорной* вероятности оценок.

Случайный вектор $\mathbf{x}$ *относится к классу* $\mathbf{C}_k$, *если*

$$F_k(\mathbf{x}) > F_j(\mathbf{x}) \text{ для всех } j \neq k, \tag{4.55}$$

где $F_k(\mathbf{x})$ *и* $F_j(\mathbf{x})$ *— элементы вектор-функции отображения*

$$\mathbf{F}(\boldsymbol{x}) = \begin{bmatrix} F_1(\boldsymbol{x}) \\ F_2(\boldsymbol{x}) \\ \ldots \\ F_M(\boldsymbol{x}) \end{bmatrix}.$$

Единственное наибольшее значение выходного сигнала существует с вероятностью 1, если соответствующие апостериорные распределения классов различаются. (Здесь предполагается использование арифметики с бесконечной точностью.) Это решающее правило имеет определенное преимущество по сравнению с моделью "отжи-

га", поскольку позволяет разделить *однозначные* (unambiguous) решения. Это значит, что вектор $\mathbf{x}$ относится к определенному классу, если соответствующее выходное значение превышает заданный порог (в логистических формах функции активации обычно используется значение 0,5), в противном случае классификация не однозначна.

В разделе 4.6 было указано, что двоичные целевые значения [0, 1], соответствующие логистической функции (4.30), на практике во время обучения сети должны измениться на небольшое значение ε, во избежание насыщения синаптических весов (в связи с далеко не бесконечной точностью представления чисел). В результате этой модификации целевые значения перестают быть двоичными, и асимптотические аппроксимации $F_k(\mathbf{x})$ не являются апостериорными вероятностями $P(\mathbf{C}_k|\mathbf{x})$ интересующих нас M классов [414]. Вместо этого $P(\mathbf{C}_k|\mathbf{x})$ линейно отображается на закрытый отрезок $[\varepsilon, 1-\varepsilon]$ так, что $P(\mathbf{C}_k|\mathbf{x}) = 0$ соответствует выходу ε, а $P(\mathbf{C}_k|\mathbf{x}) = 1 -$ выходу $1 - \varepsilon$. Так как это отображение сохраняет относительный порядок, это *не влияет* на результат применения выходного решающего правила (4.55).

Интересно также отметить следующее. Если граница решений формируется пороговым отсечением выходов многослойного персептрона относительно некоторых фиксированных значений, ее общая форма и ориентация могут быть выражены эвристически (для случая единственного скрытого слоя) в терминах количества скрытых нейронов и относительных величин связанных с ними синаптических весов [683]. Однако такой анализ не применим к границе решений, сформированной в соответствии с выходным решающим правилом (4.55). Скрытые нейроны лучше рассматривать как *нелинейные детекторы признаков* (nonlinear feature detector), призванные отобразить классы исходного входного пространства $\Re^{m_0}$ (возможно, линейно-неразделимые) в пространство активности скрытого слоя, где их линейная разделимость более вероятна.

4.8. Компьютерный эксперимент

В этом разделе с помощью компьютерного моделирования будет проиллюстрировано поведение многослойного персептронного классификатора в процессе обучения. Целью обучения является разделение двух перекрывающихся двумерных классов с гауссовым распределением, обозначенных цифрами 1 и 2. Пусть $\mathbf{C}_1$ и $\mathbf{C}_2$ — множества событий, для которых случайный вектор $\mathbf{x}$ принадлежит к классам 1 и 2 соответственно. Функцию плотности условной вероятности можно представить в следующем виде.

Для класса $\mathbf{C}_1$:

$$f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_1) = \frac{1}{2\pi\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2}\|\mathbf{x} - \mu_1\|^2\right), \tag{4.56}$$

где вектор среднего значения $\boldsymbol{\mu}_1 = [0, 0]^T$, а дисперсия $\sigma_1^2 = 1$.

Для класса $\mathbf{C}_2$:

$$f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_2) = \frac{1}{2\pi\sigma_2^2} \exp\left(-\frac{1}{2\sigma_2^2}\|\mathbf{x}-\mu_2\|^2\right), \tag{4.57}$$

где вектор среднего значения $\mu_2 = [2, 0]^T$, а дисперсия $\sigma_2^2 = 4$.

Вероятность принадлежности образа обоим классам одинакова, т.е.

$$p_1 = p_2 = 1/2.$$

На рис. 4.13 показаны трехмерные графики гауссовых распределений, определенных выражениями (4.56) и (4.57) для входного вектора $\mathbf{x}=[x_1, x_2]^T$ и размерности пространства входных сигналов $m_0 = 2$. На рис. 4.14 показаны корреляционные диаграммы классов $\mathbf{C}_1$ и $\mathbf{C}_2$ в отдельности и общая корреляционная диаграмма, отражающая суперпозицию графиков рассеяния по 500 точкам, выбранным для каждого процесса. На последней диаграмме четко видно, что распределения существенно перекрываются, а значит, высока вероятность неправильной классификации.

Байесовская граница решений

Байесовский критерий оптимальной классификации уже обсуждался в главе 3. Для данной задачи классификации на два класса предположим, что классы $\mathbf{C}_1$ и $\mathbf{C}_2$ — равновероятны, стоимость корректной классификации равна нулю и стоимости ошибок классификации одинаковы. Тогда для нахождения оптимальной границы решений можно использовать критерий отношения правдоподобия

$$\Lambda(\mathbf{x}) \underset{C_2}{\overset{C_1}{\lessgtr}} \xi, \tag{4.58}$$

где $\Lambda(\mathbf{x})$ — *отношение правдоподобия* (likelihood ratio), определяемое формулой

$$\Lambda(\mathbf{x}) = \frac{f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_1)}{f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_2)}, \tag{4.59}$$

а ξ — *порог критерия* (threshold of the test), определяемый формулой

$$\xi = \frac{p_2}{p_1} = 1. \tag{4.60}$$

В рассматриваемом примере имеем:

$$\Lambda(\mathbf{x}) = \frac{\sigma_2^2}{\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2}\|\mathbf{x}-\mu_1\|^2 + \frac{1}{2\sigma_2^2}\|\mathbf{x}-\mu_2\|^2\right).$$

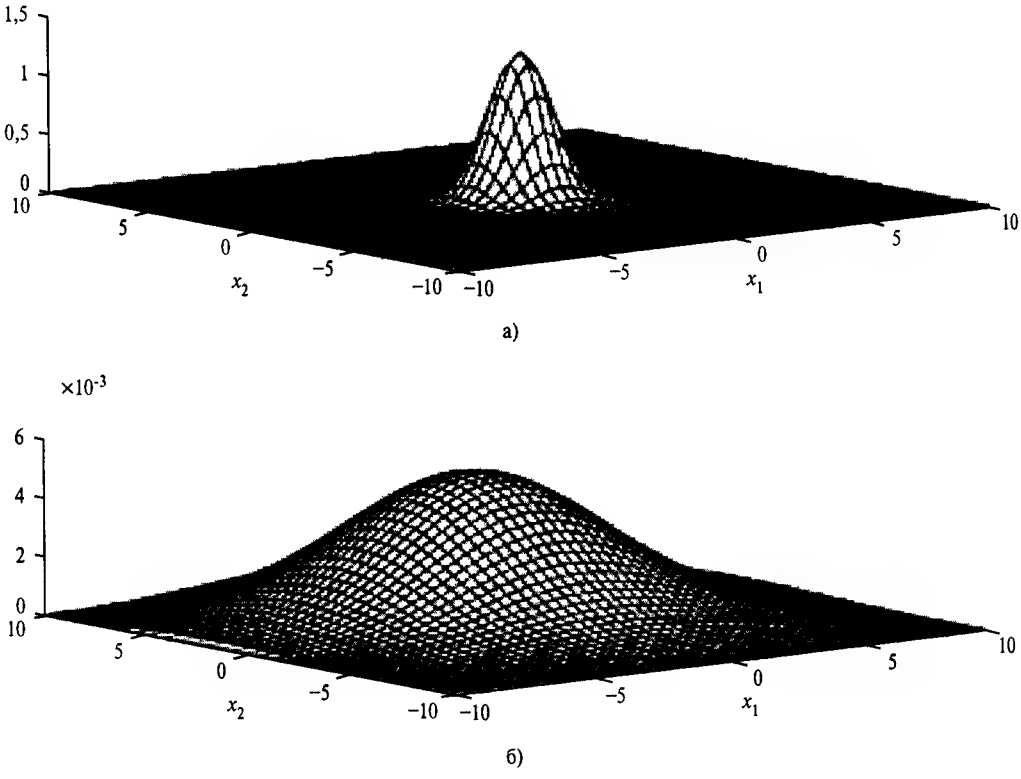


Рис. 4.13. Функции плотности вероятности: $f_x(\mathbf{x}|\mathbf{C}_1)$ (а) и $f_x(\mathbf{x}|\mathbf{C}_2)$ (б)

Таким образом, оптимальная (байесовская) граница решений определяется соотношением

$$\frac{\sigma_2^2}{\sigma_1^2}\exp\left(-\frac{1}{2\sigma_1^2}\|\mathbf{x}-\mu_1\|^2+\frac{1}{2\sigma_2^2}\|\mathbf{x}-\mu_2\|^2\right)=1$$

или

$$\frac{1}{\sigma_2^2}\|\mathbf{x}-\boldsymbol{\mu}_2\|^2-\frac{1}{\sigma_1^2}\|\mathbf{x}-\boldsymbol{\mu}_1\|^2=4\log\left(\frac{\sigma_1}{\sigma_2}\right). \tag{4.61}$$

Используя простые преобразования, оптимальную границу решений (4.61) можно переопределить в виде

$$||\mathbf{x}-\mathbf{x}||^2=r^2, \tag{4.62}$$

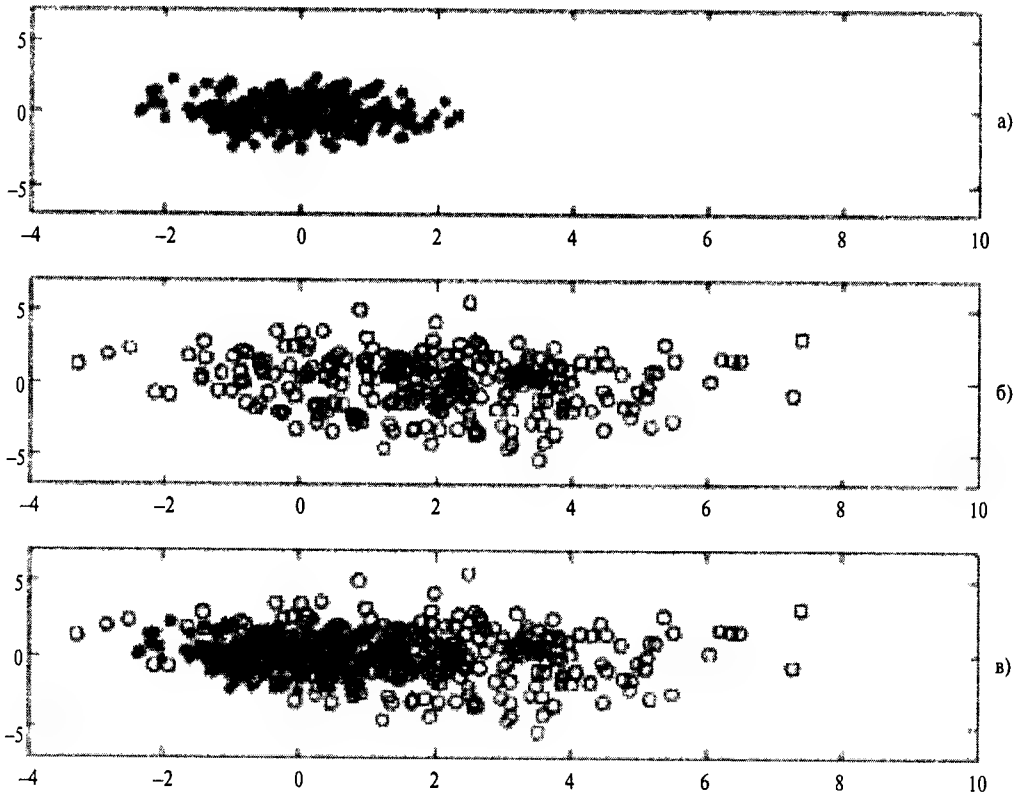


Рис. 4.14. Корреляционные диаграммы классов $\mathbf{C}_1$ (а) и $\mathbf{C}_2$ (б), а также общая корреляционная диаграмма для обоих классов (в)

где

$$\mathbf{x}_c = \frac{\sigma_2^2 \boldsymbol{\mu}_1 - \sigma_1^2 \boldsymbol{\mu}_2}{\sigma_2^2 - \sigma_1^2} \tag{4.63}$$

и

$$r^2 = \frac{\sigma_2^2 \sigma_1^2}{\sigma_2^2 - \sigma_1^2} \left[\frac{\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2}{\sigma_2^2 - \sigma_1^2} + 4 \log \left(\frac{\sigma_2}{\sigma_1} \right) \right]. \tag{4.64}$$

Уравнение (4.62) описывает окружность с центром в точке $\mathbf{x}$ и радиусом r. Обозначим область, расположенную внутри этого круга, символом Ω_1. Байесовское правило классификации для этой задачи можно сформулировать следующим образом.

Вектор наблюдения $\mathbf{x}$ *относится к классу* $\mathbf{C}_1$, *если отношение правдоподобия* $\Lambda(\mathbf{x})$ *больше порогового значения* ξ, *и к классу* $\mathbf{C}_2$ *в остальных случаях.*

В данном эксперименте центр круговой границы решений находится в точке

$$\mathbf{x}_c = \begin{bmatrix} -2/3 \\ 0 \end{bmatrix},$$

а радиус области составляет

$$r \simeq 2,34.$$

Обозначим символом c множество результатов корректной классификации, а символом e — множество результатов некорректной классификации. Тогда *вероятность ошибки* (probability of error) P_e классификатора, работающего на основе байесовского решающего правила, можем определить в виде

$$P_e = p_1 P(e|\mathbf{C}_1) + p_2 P(e|\mathbf{C}_2), \tag{4.65}$$

где $P(e|\mathbf{C}_1)$ — условная вероятность ошибки для входного вектора из класса $\mathbf{C}_1$; $P(e|\mathbf{C}_2)$ — условная вероятность ошибки для входного вектора из класса $\mathbf{C}_2$; p_1 и p_2 — *априорные* вероятности классов $\mathbf{C}_1$ и $\mathbf{C}_2$ соответственно. Для нашей задачи можно найти числовые значения вышеуказанных величин:

$$P(e|\mathbf{C}_1) \simeq 0,1056,$$

$$P(e|\mathbf{C}_2) \simeq 0,2642.$$

Так как классы равновероятны, т.е. $p_1 = p_2 = 1/2$, то

$$P_e \simeq 0,1849.$$

Следовательно, *вероятность правильной классификации* (probability of correct classification) составляет

$$P = 1 - P_e \simeq 0,8151.$$

Экспериментальное построение оптимального многослойного персептрона

В табл. 4.1 приведены параметры многослойного персептрона (multilayer perceptron — MLP) с одним слоем скрытых нейронов, который обучается с помощью алгоритма

ТАБЛИЦА 4.1. Переменные параметры многослойного персептрона

Параметр	*Символ*	*Типичный диапазон*
Количество скрытых нейронов	m_1	$(2, \infty)$
Параметр скорости обучения	η	$(0, 1)$
Константа момента	α	$(0, 1)$

ТАБЛИЦА 4.2. Результаты моделирования для двух скрытых нейронов (коэффициент скорости обучения равен 0,1, фактор момента — 0)

Количество проходов	*Размер обучающего множества*	*Количество эпох*	*Среднеквадратическая ошибка*	*Вероятность корректной классификации* $P_c, \%$
1	500	320	0,2375	80,36
2	2000	80	0,2341	80,33
3	8000	20	0,2244	80,47

обратного распространения в последовательном режиме. Поскольку единственной целью классификатора является достижение приемлемого уровня корректной классификации, этот критерий и применяется для проверки оптимальности переменных параметров MLP.

Оптимальное число скрытых нейронов

Учитывая практический подход к задаче определения оптимального количества скрытых нейронов (m_1), будем использовать следующий критерий. Необходимо найти минимальное количество скрытых нейронов, которое обеспечивает производительность, близкую (в пределах 1%) к байесовскому. Исходя из этого, эксперимент начинается с двух скрытых нейронов. Результаты моделирования приведены в табл. 4.2.

Поскольку целью первого этапа моделирования является проверка достаточности или двух скрытых нейронов, параметрам α и η произвольно присвоены некоторые номинальные значения. Для каждого запуска процесса моделирования генерировалось множество примеров обучения, с равной вероятностью содержащее образы классов $\mathbf{C}_1$ и $\mathbf{C}_2$ с гауссовым распределением. Примеры из этого множества последовательно подавались на вход сети в течение нескольких циклов, или *эпох* (epoch). Количество эпох выбиралось так, чтобы общее число обучающих примеров в процессе обучения оставалось постоянным. Таким образом, можно усреднить результаты для обучающих множеств разного размера.

В табл. 4.2 и последующих таблицах *среднеквадратическая ошибка* вычислялась в соответствии с функционалом, определенным формулой (4.53). Необходимо под-

черкнуть, что для этих таблиц определялась именно среднеквадратическая ошибка, хотя *минимум среднеквадратической ошибки не всегда отражает хороший уровень обобщения* (т.е. хорошую производительность на ранее не использованных данных).

В результате обучения сети на N примерах вероятность корректной классификации теоретически определяется выражением

$$P(,N) = p_1 P(,N|\mathbf{C}_1) + p_2 P(,N|\mathbf{C}_2), \tag{4.66}$$

где $p_1 = p_2 = 1/2$ и

$$P(c,N|\mathbf{C}_1) = \int_{\Omega_1(N)} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_1)d\mathbf{x}, \tag{4.67}$$

$$P(c,N|\mathbf{C}_2) = 1 - \int_{\Omega_1(N)} f_{\mathbf{X}}(\mathbf{x}|\mathbf{C}_2)d\mathbf{x} \tag{4.68}$$

и $\Omega_1(N)$ — область пространства решений, в которой многослойный персептрон (обученный на N примерах) относит вектор $\mathbf{x}$ (представляющий реализацию случайного вектора $\mathbf{X}$) к классу $\mathbf{C}_1$. Эту область обычно находят экспериментально, оценивая функцию отображения, которой обучалась сеть, и применяя выходное решающее правило (4.55). К сожалению, численно оценить величины$P(,N|\mathbf{C}_1)$ и $P(,N|\mathbf{C}_2)$ непросто, так как сложно отыскать формальное определение границы решений $\Omega_1(N)$.

Поэтому воспользуемся экспериментальным подходом и протестируем обученный многослойный персептрон на независимом множестве примеров, которые снова случайно сгенерируем на основе гауссовых распределений классов $\mathbf{C}_1$ и $\mathbf{C}_2$ с равной вероятностью. Введем случайную переменную A, которая означает количество корректно классифицированных примеров из множества мощности N. Тогда частное

$$p_N = A/N$$

является случайной величиной, которая обеспечивает максимально правдоподобную несмещенную оценку реальной эффективности классификации p. Предполагая, что p является константой для N пар "вход-выход", можно использовать *предел Чернова* [257]:

$$P(|p_N - p| > \varepsilon) < 2\exp(-2\varepsilon^2 N) = \delta.$$

Применяя предел Чернова, получим $N = 26500$ для $\delta = 0,01$ и $\varepsilon = 0,01$ (т.е. с достоверностью 99% можно утверждать, что оценка p характеризуется данным допустимым отклонением). Рассмотрим тестовое множество, содержащее $N = 32000$ образов. В последнем столбце табл. 4.2 показана вероятность корректной классифи-

ТАБЛИЦА 4.3. Результаты моделирования для четырех скрытых нейронов ($\eta = 0.1, \alpha = 0$)

Количество проходов	*Размер обучающего множества*	*Количество эпох*	*Среднеквадратическая ошибка*	*Вероятность корректной классификации* P_c, %
1	500	320	0,2129	80,80
2	2000	80	0,2108	80,81
3	8000	20	0,2142	80,19

кации для тестового множества такой мощности, усредненная по десяти независимым экспериментам.

Эффективность классификации многослойного персептрона с двумя скрытыми нейронами (см. табл. 4.2) достаточно близка к производительности байесовского классификатора, равной P_c =81,51%. Отсюда логично заключить, что рассматриваемую задачу классификации можно решить с помощью многослойного персептронного классификатора с двумя скрытыми нейронами. Чтобы подтвердить это заключение, в табл. 4.3 представлены результаты моделирования для четырех скрытых нейронов (остальные параметры остались без изменений). Хотя среднеквадратическая ошибка в табл. 4.3 для четырех скрытых нейронов несколько ниже, чем в табл. 4.2 для двух скрытых нейронов, средний уровень корректной классификации практически не улучшился — в одном из тестов результаты оказались даже несколько хуже. Поэтому продолжим вычислительный эксперимент для двух скрытых нейронов.

Оптимальное обучение и константа момента

Для оценки оптимальности значений параметров скорости обучения η и момента α можно использовать любое из трех следующих определений.

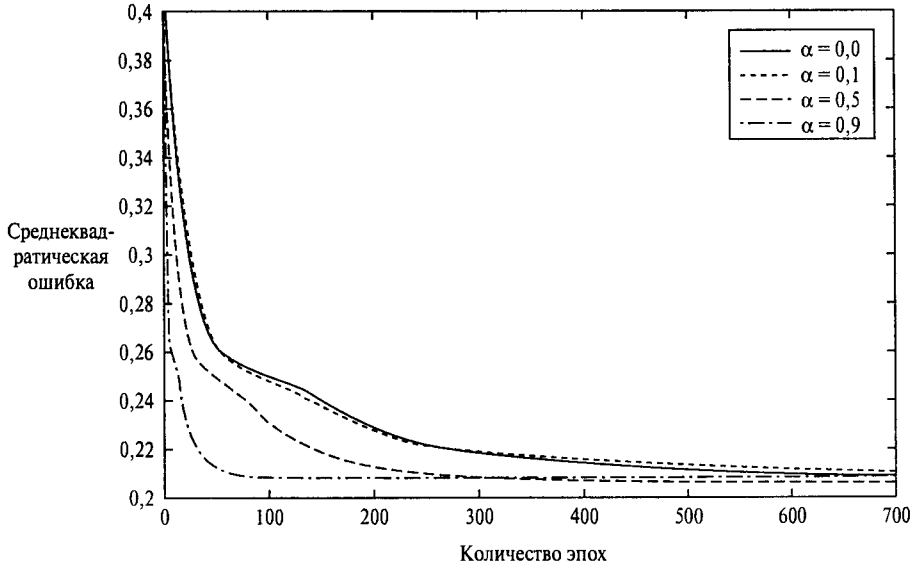
1. Оптимальными считаются константы α и η, при которых сходимость сети к локальному минимуму на поверхности ошибок достигается в среднем за минимальное количество эпох.
2. Оптимальными считаются константы α и η, при которых сходимость сети к глобальному минимуму на поверхности ошибок (в наихудшем случае или в среднем) достигается за минимальное число эпох.
3. Оптимальными считаются константы α и η, при которых средний показатель сходимости к конфигурации, имеющей наилучшие обобщающие способности во всем пространстве входных сигналов, достигается за минимальное количество эпох.

Используемые здесь термины "средний" и "наихудший" означают распределение обучающих пар типа "вход-выход". Определение 3 является наиболее ценным для практики, однако его сложно применить, так как минимизация среднеквадратической ошибки — это математический критерий оптимальности, применяемый при обучении сети, и, как говорилось ранее, снижение среднеквадратической ошибки не всегда ведет к улучшению обобщающей способности. С исследовательской точки зрения второе определение представляет больший интерес, нежели первое. Например, в [684] были представлены результаты серьезного исследования оптимальной настройки параметра скорости обучения η, при котором многослойному персептрону требуется минимальное количество эпох для аппроксимации глобально оптимальной матрицы синаптических весов с заданной точностью (правда, только для частного случая линейных нейронов). Тем не менее в общем случае при экспериментальном и эвристическом подходе определения оптимальных значений η и α используется определение 1. Поэтому в своем эксперименте будем руководствоваться именно этим определением.

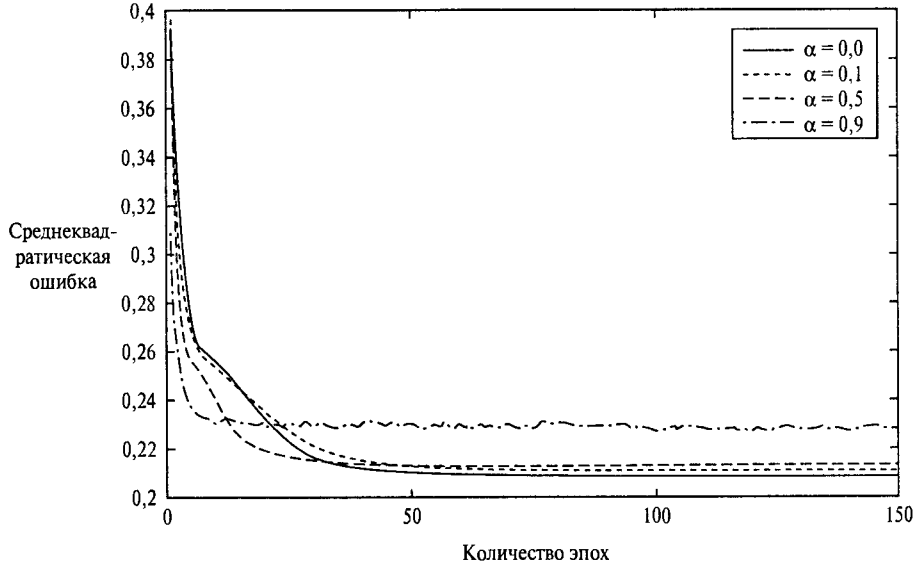
Используя многослойный персептрон с двумя скрытыми нейронами, а также различные комбинации значений параметра скорости обучения $\eta \in \{0.01, 0.1, 0.5, 0.9\}$ и константы момента $\alpha \in \{0.0, 0.1, 0.5, 0.9\}$, исследуем скорость сходимости сети. Каждая из конфигураций обучается на одном и том же множестве примеров при одном и том же наборе исходных значений синаптических весов для $N = 500$. Поэтому результаты можно сравнивать напрямую, без внесения поправок. Процесс обучения длился в течение 700 эпох. Такую продолжительность обучения мы посчитали достаточной для достижения алгоритмом обратного распространения некоторого локального минимума на поверхности ошибок. Усредненные кривые процесса обучения показаны на рис. 4.15, *а–г* для различных значений параметра η.

Показанные экспериментальные кривые отражают следующие тенденции обучения.

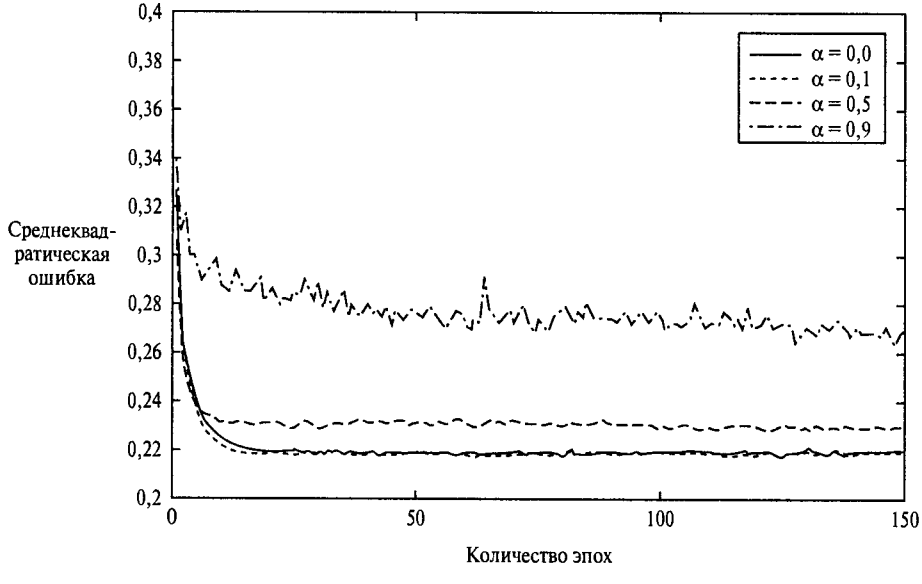
- В общем случае малые значения параметра η обеспечивают более медленную сходимость. При этом более точно определяется точка локального минимума на поверхности ошибок. Это интуитивно понятно, так как при меньших значениях η поиск минимума выполняется в более широкой области поверхности ошибок.
- При $\eta \rightarrow 0$ выбор больших значений константы момента ($\alpha \rightarrow 1$) приводит к увеличению скорости сходимости. С другой стороны, при $\eta \rightarrow 1$ малые значения фактора момента $\alpha \rightarrow 0$ повышают устойчивость обучения.
- Использование значений $\eta = \{0{,}5;\ 0{,}9\}$ и $\alpha = 0{,}9$ приводит к колебаниям среднеквадратической ошибки в процессе обучения и более высокому ее значению по завершении процесса сходимости. И то и другое — нежелательно.


0,4
0,38
0,36
0,34
0,32
0,3
0,28
0,26
0,24
0,22
0,2
Среднеквадратическая ошибка
0
100
200
300
400
500
600
700
Количество эпох
α = 0,0
α = 0,1
α = 0,5
α = 0,9

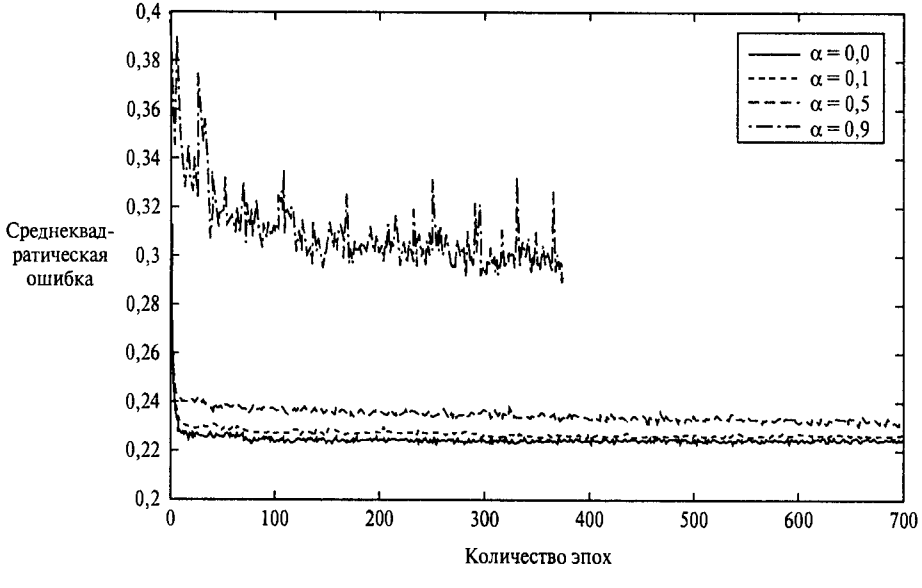
а)


0,4
0,38
0,36
0,34
0,32
0,3
0,28
0,26
0,24
0,22
0,2
Среднеквадратическая ошибка
0
50
100
150
Количество эпох
α = 0,0
α = 0,1
α = 0,5
α = 0,9

б)



в)



г)

Рис. 4.15. Усредненные кривые обучения для различных значений константы момента α и параметра скорости обучения: $\eta = 0,01$ (а); $\eta = 0,1$ (б); $\eta = 0,5$ (в); $\eta = 0,9$ (г)

ТАБЛИЦА 4.4. Конфигурация оптимизированного многослойного персептрона

Параметр	*Символ*	*Значение*
Оптимальное число скрытых нейронов	m_{opt}	2
Оптимальный параметр скорости обучения	η_{opt}	0,1
Оптимальная константа момента	α_{opt}	0,5

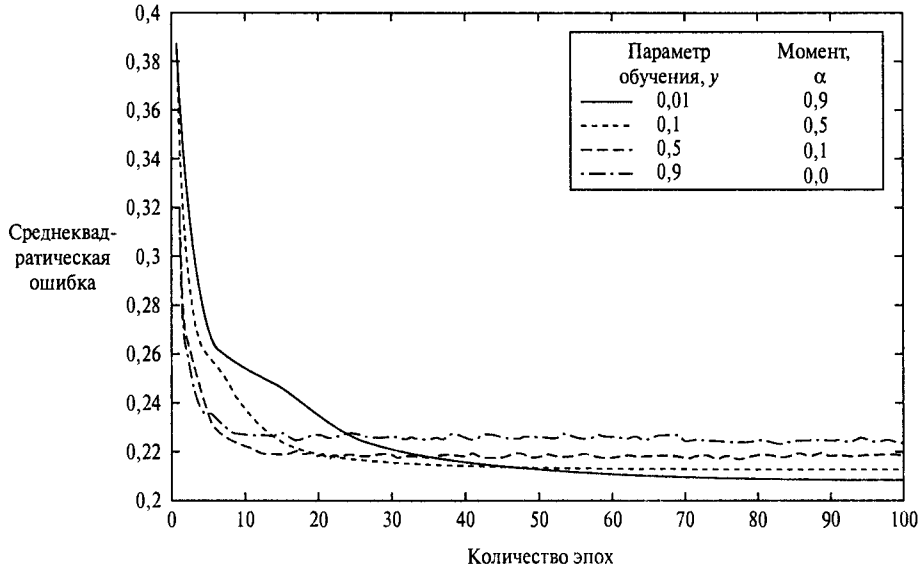


Рис. 4.16. Лучшие кривые обучения, выбранные на рис. 4.15

На рис. 4.16 показаны графики "наилучших" кривых обучения по каждой из групп, представленных на рис. 4.15, для выбора наилучшей кривой в смысле определения 1. На рисунке видно, что оптимальное значение параметра скорости обучения η_{opt} составляет порядка 0,1, а α_{opt} — около 0,5. В табл. 4.4 приведены оптимальные значения параметров сети, которые будут использоваться в оставшейся части эксперимента. Как видно на рис. 4.16, конечная среднеквадратическая ошибка мало отличается для различных кривых. Это значит, что поверхность ошибок в нашей задаче достаточно гладкая.

Оценка оптимальной архитектуры сети

Для "оптимизированного" многослойного персептрона, параметры которого приведены в табл. 4.4, была исследована граница решений, построена усредненная по ансамблю кривая обучения и оценена вероятность корректной классификации. Если обучающее множество конечно, то функциональное преобразование сети, обученной

ТАБЛИЦА 4.5. Обобщенные статистические показатели производительности

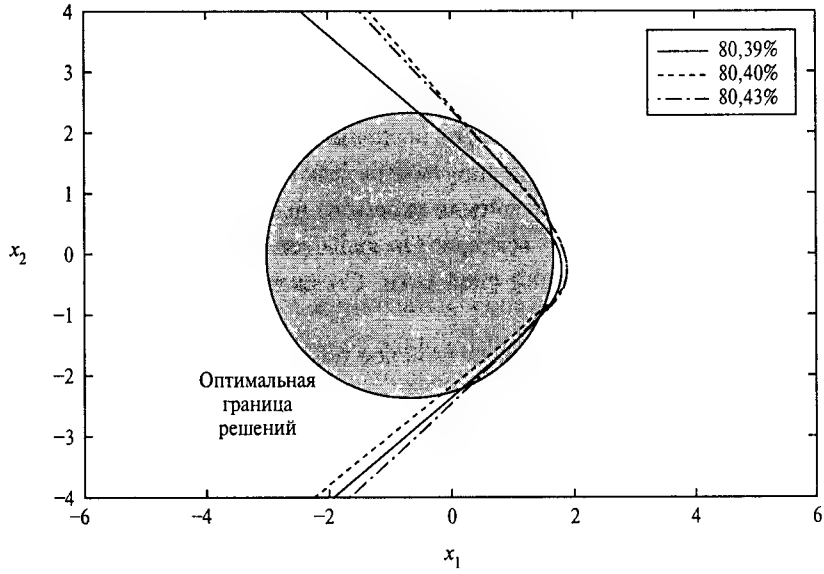
Показатель производительности	*Среднее значение*	*Стандартное отклонение*
Вероятность корректной классификации	79,70%	0,44%
Окончательная среднеквадратическая ошибка	0,2277	0,0118

при оптимальных значениях параметров, стохастично по своей природе. Поэтому показатели производительности усреднялись по двадцати независимо обучаемым сетям. Каждое из обучающих множеств состояло из 1000 примеров, выбранных из классов $\mathbf{C}_1$ и $\mathbf{C}_2$ с одинаковой вероятностью. Эти примеры предъявлялись сетям в случайной последовательности. Как и ранее, обучение выполнялось в течение 700 эпох. Для экспериментального определения вероятности корректной классификации использовалось сгенерированное ранее тестовое множество из 32000 примеров. На рис. 4.17, *а* показаны три "лучшие" границы решений для трех из 20 сетей. На рис. 4.17, *б* показаны три "наихудшие" границы решений для трех других сетей из этой же группы. На этих рисунках видно, что границы решений, построенные с помощью алгоритма обратного распространения, являются выпуклыми по отношению к области классификации вектора наблюдения $\mathbf{x}$.

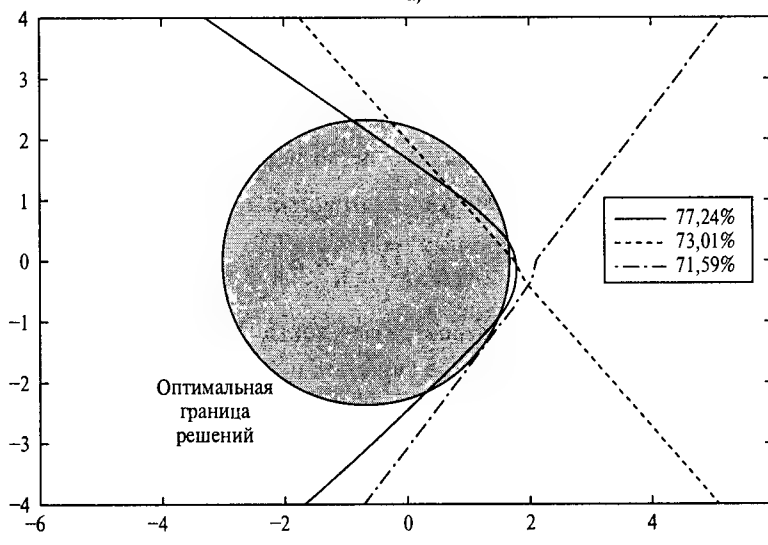
Статистические данные о производительности, вероятности корректной классификации и конечной среднеквадратической ошибке, вычисленной для обучающего множества, приведены в табл. 4.5. Напомним, что вероятность корректной классификации для оптимального байесовского классификатора составляет 81,51%.

4.9. Извлечение признаков

Скрытые нейроны (hidden neuron) играют крайне важную роль в работе многослойного персептрона, обучаемого методом обратного распространения, поскольку они выступают в роли *детекторов признаков* (feature detector). В ходе обучения скрытые нейроны постепенно "выявляют" характерные черты данных обучения. Это осуществляют с помощью нелинейного преобразования входных данных в новое, *скрытое пространство* (hidden space) (или *пространство признаков* (feature space)). Эти термины будут использоваться во всей книге. В новом пространстве классы (если взять для примера задачу классификации) легче отделить друг от друга, чем в исходном пространстве. Это утверждение уже было проиллюстрировано на примере решения задачи XOR в разделе 4.5.



а)



б)

Рис. 4.17. Графики трех "лучших" (а) и трех "наихудших" (б) границ решений с точностью классификации 80,39, 80,40, 80,43, и 77,24%, 73,01 и 71,59% соответственно

Чтобы перевести обсуждение в математический контекст, рассмотрим многослойный персептрон с одним слоем, состоящим из m_1 нелинейных скрытых нейронов, и одним слоем линейных выходных нейронов размерности $m_2 = M$. Выбор линейных нейронов в выходном слое обусловлен необходимостью сфокусировать внимание только на роли скрытых нейронов в работе многослойного персептрона. Синаптические веса сети нужно настроить так, чтобы минимизировать среднеквадратическую ошибку (по N образам) между целевым выходом (желаемым откликом) и фактическим выходным сигналом сети, генерируемым в ответ на m_0-мерный вектор входного сигнала. Пусть $z_j(n)$ — выходной сигнал скрытого нейрона j, генерируемый в ответ на представление n-го входного вектора. Он является нелинейной функцией образа (вектора), подаваемого на входной слой сети. С каждым нейроном связана сигмоидальная функция активации.

Выходной сигнал нейрона k выходного слоя описывается выражением

$$y_k(n) = \sum_{j=0}^{m_1} w_{kj} z_j(n), \quad k = 1, 2, \ldots, M; \quad n = 1, 2, \ldots, N, \tag{4.69}$$

где w_{k0} — смещение, связанное с нейроном k. Функция стоимости, которую следует минимизировать, записывается в виде

$$\mathbf{E}_{\text{av}} = \frac{1}{2N} \sum_{n=1}^{N} \sum_{k=1}^{M} (d_k(n) - y_k(n))^2. \tag{4.70}$$

Заметим, что здесь подразумевается использование пакетного режима обучения. Используя (4.68) и (4.70), можно довольно легко переписать функцию стоимости в более компактной матричной форме:

$$\mathbf{E}_{\text{av}} = \frac{1}{2N} \left\| \tilde{\mathbf{D}} - \mathbf{W}\tilde{\mathbf{Z}} \right\|^2, \tag{4.71}$$

где $\mathbf{W}$ — матрица синаптических весов размерности $M \times m_1$, относящаяся к выходному слою сети. Матрица $\tilde{\mathbf{Z}}$ выходов скрытых нейронов (за вычетом их среднего значения) имеет размерность $m_1 \times N$. Эта матрица состоит из сигналов, полученных в результате обработки N входных образов (входных сигналов), т.е.

$$\tilde{\mathbf{Z}} = \{(z_j(n) - \mu_{z_j}); \; j = 1, 2, \ldots, m_1; \; n = 1, 2, \ldots, N\},$$

где μ_{z_j} — среднее значение сигнала $z_j(n)$. Соответственно матрица $\tilde{\mathbf{D}}$ желаемых откликов размерности $M \times N$, представляемых выходному слою сети, имеет вид

$$\tilde{\mathbf{D}} = \{(d_k(n) - \mu_{d_k}); \; k = 1, 2, \ldots, M; \; n = 1, 2, \ldots, N\},$$

где μ_{dk} — среднее значение $d_k(n)$. Минимизация функции $\mathbf{E}_{\text{av}}$, определяемой выражением (4.70), — это линейная задача, решение которой описывается уравнением

$$\mathbf{W} = \tilde{\mathbf{D}}\tilde{\mathbf{Z}}^{+}, \tag{4.72}$$

где $\tilde{\mathbf{Z}}^{+}$ — псевдообратная матрица для $\tilde{\mathbf{Z}}$. Минимальное значение функции $\mathbf{E}_{\text{av}}$ можно представить в виде (см. упражнение 4.7)

$$\mathbf{E}_{\text{av,min}} = \frac{1}{2N}\text{tr}\left[\tilde{\mathbf{D}}\tilde{\mathbf{D}}^{T} - \tilde{\mathbf{D}}\tilde{\mathbf{Z}}^{T}\left(\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^{T}\right) + \tilde{\mathbf{Z}}\tilde{\mathbf{D}}^{T}\right], \tag{4.73}$$

где tr[·] — оператор следа. Так как целевые образы, представленные матрицей $\tilde{\mathbf{D}}$, фиксированы, задача минимизации функции стоимости $\mathbf{E}_{\text{av}}$ относительно синаптических весов многослойного персептрона эквивалентна максимизации *дискриминантной функции* [1120]:

$$\mathbf{D} = \text{tr}[\mathbf{C}_b\mathbf{C}_t^{+}], \tag{4.74}$$

где матрицы $\mathbf{C}_b$ и $\mathbf{C}_t$ определяются следующим образом.

- $\mathbf{C}_t$ — $m_1 \times m_1$-*матрица общей ковариации* (total covariance matrix) выходов скрытых нейронов при представлении N входных сигналов

$$\mathbf{C}_t = \tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^{T}. \tag{4.75}$$

  Матрица $\mathbf{C}_t^{+}$ является псевдообратной по отношению к матрице $\mathbf{C}_t$.
- Матрица $\mathbf{C}_b$ размерности $m_1 \times m_1$ определяется выражением

$$\mathbf{C}_b = \tilde{\mathbf{Z}}\tilde{\mathbf{D}}^{T}\tilde{\mathbf{D}}\tilde{\mathbf{Z}}^{T}. \tag{4.76}$$

Обратите внимание, что дискриминантная функция $\mathbf{D}$ согласно (4.74) определяется исключительно скрытыми нейронами многослойного персептрона. Заметим также, что количество скрытых слоев, участвующих в нелинейном преобразовании и генерации дискриминантной функции, не ограничено. Если многослойный персептрон содержит несколько скрытых слоев, то матрица $\tilde{\mathbf{Z}}$ описывает все множество образов (сигналов) в пространстве, определенном последним слоем скрытых нейронов.

Чтобы интерпретировать матрицу $\mathbf{C}_b$, рассмотрим частный случай *схемы кодирования "один из M"* (one-from-M coding scheme) [1120]. Это значит, что k-й элемент целевого вектора (желаемого отклика) равен единице, если входной сигнал принадлежит классу k, и нулю в противном случае:

$$d(n) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow k\text{-й элемент, } d(n) \in \mathbf{C}_k.$$

Таким образом, для M классов $\mathbf{C}_k, k = 1, 2, \ldots, M$, к каждому из которых относится N_k образов, таких, что

$$\sum_{k=1}^{M} N_k = N,$$

матрицу $\mathbf{C}_b$ для данной схемы кодирования можно представить в виде

$$\mathbf{C}_b = \sum_{k=1}^{M} N_k^2 (\boldsymbol{\mu}_{z,k} - \boldsymbol{\mu}_z)(\boldsymbol{\mu}_{z,k} - \boldsymbol{\mu}_z)^T, \tag{4.77}$$

где вектор $\boldsymbol{\mu}_{z,k}$ размерности $m_1 \times 1$ является средним значением вектора выходов скрытых нейронов для N входных образов. В соответствии с (4.77) матрицу $\mathbf{C}_b$ можно интерпретировать как *матрицу взвешенной межклассовой ковариации* (weighted between-class covariance matrix) выходных сигналов скрытого слоя.

Таким образом, для схемы кодирования "один из M" многослойный персептрон максимизирует дискриминантную функцию, представляющую собой след произведения двух величин: матрицы взвешенной межклассовой ковариации и псевдообратной матрицы для матрицы общей ковариации. Из этого примера видно, что при обучении многослойного персептрона методом обратного распространения в качестве *предварительной информации* учитываются пропорции образов в рамках отдельных классов.

Связь с линейным дискриминантом Фишера

Дискриминантная функция, определенная в (4.74), характерна лишь для многослойного персептрона. Однако она тесно связана с *линейным дискриминантом Фишера* (Fisher's linear discriminant), который описывает линейное преобразование многомерной задачи в одномерную. Рассмотрим переменную y, представляющую собой линейную комбинацию элементов входного вектора $\mathbf{x}$. Более точно, пусть y является скалярным произведением вектора $\mathbf{x}$ и вектора настраиваемых весов $\mathbf{w}$ (содержащего в качестве первого элемента порог b):

$$y = \mathbf{w}^T \mathbf{x}.$$

Вектор $\mathbf{x}$ выбирается из множества $\mathbf{C}_1$ или $\mathbf{C}_2$, которые, в свою очередь, отличаются векторами средних значений ($\boldsymbol{\mu}_1$ и $\boldsymbol{\mu}_2$ соответственно). *Критерий Фишера* (Fisher's criterion), определяющий степень различия между двумя классами, задается следующим образом:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{C}_b \mathbf{w}}{\mathbf{w}^T \mathbf{C}_t \mathbf{w}},$$

где $\mathbf{C}_b$ — *матрица межклассовой ковариации* (between-class covariance matrix),

$$\mathbf{C}_b = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T,$$

а $\mathbf{C}_t$ — общая *матрица внутриклассовой ковариации* (within-class covariance matrix),

$$\mathbf{C}_t = \sum_{n \in \mathbf{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T + \sum_{n \in \mathbf{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T.$$

Матрица внутриклассовой ковариации $\mathbf{C}_t$ пропорциональна матрице ковариации обучающего множества. Это симметричная и неотрицательно определенная матрица, которая обычно является несингулярной, если размер множества обучения достаточно велик. Матрица межклассовой ковариации $\mathbf{C}_b$ также является симметричной и неотрицательно определенной, однако она сингулярна. Следует отметить, что матричное произведение $\mathbf{C}_b\mathbf{w}$ всегда направлено в сторону вектора разности между средними значениями $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$. Это свойство непосредственно следует из определения матрицы $\mathbf{C}_b$.

Выражение, определяющее критерий Фишера $J(\mathbf{w})$, называют *обобщенным фактором Рэлея* (generalized Rayleigh quotient). Вектор $\mathbf{w}$, максимизирующий эту величину, должен удовлетворять условию

$$\mathbf{C}_b\mathbf{w} = \lambda \mathbf{C}_t\mathbf{w}. \tag{4.78}$$

Уравнение (4.78) описывает обобщенную задачу нахождения собственных чисел. В нашем случае матричное произведение $\mathbf{C}_b\mathbf{w}$ всегда направлено в сторону разности $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$, так что уравнение (4.78) достаточно легко разрешить относительно $\mathbf{w}$:

$$\mathbf{w} = \mathbf{C}_t^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2). \tag{4.79}$$

Это решение называется *линейным дискриминантом Фишера* (Fisher's linear discriminant) [269]. Возвращаясь к вопросу извлечения признаков, вспомним, что дискриминантная функция $\mathbf{D}$, определяемая формулой (4.74), связывает матрицы межклассовой и общей ковариации образов, трансформированных в пространство скрытых нейронов сети. Дискриминантная функция $\mathbf{D}$ аналогична линейному дискриминанту Фишера. Именно поэтому нейронные сети так хорошо решают задачу классификации.

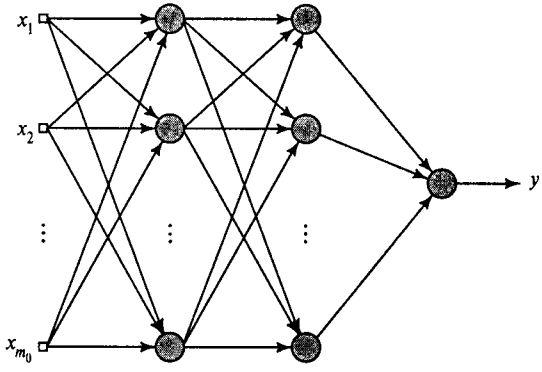


Рис. 4.18. Многослойный персептрон с двумя скрытыми слоями и одним выходным нейроном

4.10. Обратное распространение ошибки и дифференцирование

Метод обратного распространения ошибки является специфической реализацией *градиентного спуска* (gradient descent) в пространстве весов многослойных сетей прямого распространения. Основная идея этого метода заключается в эффективном вычислении *частных производных* (partial derivative) функции сети $F(\mathbf{w}, \mathbf{x})$ по всем элементам настраиваемого вектора весов $\mathbf{w}$ для данного входного вектора $\mathbf{x}$. В этом и заключается вычислительная мощность алгоритма обратного распространения[5].

Для большей конкретизации рассмотрим многослойный персептрон с входным слоем, состоящим из m_0 узлов, двумя скрытыми слоями и одним выходным нейроном (рис. 4.18). Элементы вектора весов $\mathbf{w}$ упорядочены по слоям (начиная с первого скрытого), затем по нейронам и, наконец, по синаптическим связям каждого нейрона. Пусть $w_{ji}^{(l)}$ — синаптический вес, связывающий нейрон i с нейроном j слоя $l = 0, 1, \ldots$. Для первого скрытого слоя ($l = 1$) индекс i относится не к нейрону, а к входному узлу. Для $l = 3$, что соответствует выходному слою, $j = 1$. Требуется вычислить производные функции $F(\mathbf{w}, \mathbf{x})$ по всем элементам вектора весов $\mathbf{w}$ для заданного входного вектора $\mathbf{x} = [x_1, x_2, \ldots, x_{m_0}]^T$. Заметим, что для $l = 2$ (т.е. для второго скрытого слоя) функция $F(\mathbf{w}, \mathbf{x})$ имеет форму, аналогичную правой части выражения (4.69). Вектор весов $\mathbf{w}$ включен в список аргументов функции $F(\mathbf{w}, \mathbf{x})$, чтобы привлечь к нему внимание.

---

[5] Первое документированное описание обратного распространения ошибки для эффективной оценки градиента было предложено в [1128]. Материал, представленный в разделе 4.10, соответствует подходу, предложенному в [919]. Более полно этот вопрос освещен в [1126].

Многослойный персептрон на рис. 4.18 определяется *архитектурой* **A** (представляющей собой дискретный параметр) и *вектором весов* **w** (составленным из вещественных элементов). Пусть $\mathbf{A}_j^{(l)}$ — часть архитектуры, включающая в себя фрагмент нейронной сети от входного слоя (l =0) до узла j слоя $l(l = 1, 2, 3)$. Соответственно можно записать

$$F(\mathbf{w}, \mathbf{x}) = \varphi(\mathbf{A}_1^{(3)}), \tag{4.80}$$

где φ — функция активации. Однако $\mathbf{A}_1^{(3)}$ необходимо интерпретировать исключительно как символ обозначения архитектуры, а не переменную. Таким образом, адаптируя выражения (4.1), (4.2), (4.11) и (4.23) к данной ситуации, получим следующий результат:

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial w_{1k}^{(3)}} = \varphi'(\mathbf{A}_1^{(3)})\varphi(\mathbf{A}_k^{(2)}), \tag{4.81}$$

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial w_{kj}^{(2)}} = \varphi'(\mathbf{A}_1^{(3)})\varphi'(\mathbf{A}_k^{(2)})\varphi(\mathbf{A}_j^{(1)})w_{1k}^{(3)}, \tag{4.82}$$

$$\frac{\partial F(\mathbf{w}, \mathbf{x})}{\partial w_{ji}^{(1)}} = \varphi'(\mathbf{A}_1^{(3)})\varphi'(\mathbf{A}_j^{(1)})\mathbf{x}_i \left[\sum_k w_{1k}^{(3)}\varphi'(\mathbf{A}_k^{(2)})w_{kj}^{(2)}\right], \tag{4.83}$$

где φ' — частная производная нелинейной функции φ по своим аргументам; x_i — i-й элемент входного вектора **x**. Аналогично можно вывести выражения для частных производных любой общей сети с большим числом слоев и выходных нейронов.

Выражения (4.81)–(4.83) являются основой для вычисления степени чувствительности функции сети $F(\mathbf{w}, \mathbf{x})$ к вариациям элементов вектора весов **w**. Пусть ω — некоторый элемент вектора весов **w**. *Чувствительность* (sensitivity) $F(\mathbf{w}, \mathbf{x})$ к элементу ω формально определяется выражением

$$S_\omega^F = \frac{\partial F/F}{\partial \omega/\omega}, \omega \in \mathbf{w}.$$

Именно по этой причине нижнюю часть графа передачи сигнала на рис. 4.7 мы назвали *графом чувствительности* (sensitivity graph).

Матрица якобиана

Пусть W — общее количество свободных параметров (т.е. синаптических весов и порогов) многослойного персептрона, которые в упорядоченном указанным выше способом формируют вектор весов **w**. Пусть N — общее количество примеров, использованных для обучения сети. Применяя метод обратного распространения, можно вычислить множество W частных производных аппроксимирующей функции F[**w**,

$\mathbf{x}(n)]$ по элементам вектора весов $\mathbf{w}$ для каждого примера $\mathbf{x}(n)$ из обучающего множества. Повторяя эти вычисления для $n = 1, 2, \ldots, N$, можно получить матрицу частных производных размерности $N \times W$. Эта матрица называется *якобианом* J многослойного персептрона, вычисленным в точке $\mathbf{x}(n)$. Каждая строка якобиана соответствует одному примеру из обучающего множества.

Экспериментально подтверждается, что многие задачи обучения нейросетей внутренне *плохо обусловлены* (ill-conditioned), что ведет к неполноте ранга якобиана J [920]. *Ранг* (rank) матрицы равен количеству ее линейно-независимых строк или столбцов (наименьшему из них). Ранг якобиана считается *неполным* (rank deficient), если он меньше значения $\min(N, W)$. Неполнота ранга якобиана приводит к тому, что алгоритм обратного распространения получает только частичную информацию о возможных направлениях поиска, что, в свою очередь, значительно удлиняет время обучения.

4.11. Гессиан

Гессиан, или *матрица Гессе* (Hessian matrix), функции стоимости $\mathbf{E}_{\text{av}}(\mathbf{w})$, обозначаемая символом $\mathbf{H}$, определяется как вторая производная $\mathbf{E}_{\text{av}}(\mathbf{w})$ по вектору весов $\mathbf{w}$:

$$\mathbf{H} = \frac{\partial^2 \mathbf{E}_{\text{av}}(\mathbf{w})}{\partial \mathbf{w}^2}. \tag{4.84}$$

Гессиан играет важную роль в изучении нейронных сетей; в частности, можно отметить следующее[6].

1. Собственные числа Гессиана оказывают определяющее влияние на динамику обучения методом обратного распространения.
2. На основе матрицы, обратной Гессиану, можно выделить несущественные синаптические веса многослойного персептрона и отключить их. Этот вопрос детально обсуждается в разделе 4.15.
3. Гессиан составляет основу методов оптимизации второго порядка, применяемых в качестве альтернативы алгоритма обратного распространения (см. раздел 4.18).

---

[6] К другим аспектам создания нейронных сетей на основе матрицы Гессе относятся следующие [130].

Вычисление Гессиана составляет основу процедуры повторного обучения многослойного персептрона после внесения небольших изменений во множество примеров обучения.

В контексте байесовского обучения необходимо отметить следующее.

Матрицу, обратную Гессиану, можно использовать для определения границ ошибок при нелинейном прогнозировании, выполняемом с помощью обученной нейронной сети.

Собственные значения матрицы Гессе можно использовать для нахождения значений параметров регуляризации.

Итеративная процедура вычисления[7] Гессиана описывается в разделе 4.15. Сейчас обратим внимание на п. 1.

В главе 3 уже говорилось о том, что собственные числа Гессиана определяют свойства сходимости алгоритма LMS. То же можно сказать и об алгоритме обратного распространения, однако механизм здесь более сложный. Обычно матрица Гессе для поверхности ошибок многослойного персептрона, обучаемого по алгоритму обратного распространения, имеет следующие особенности распределения собственных чисел [617], [625].

- Небольшое количество собственных чисел с малыми значениями.
- Большое количество собственных чисел со средними значениями.
- Малое количество больших собственных чисел.

Факторы, на которые влияет такой ход вещей, можно сгруппировать следующим образом.

- Входные и выходные сигналы нейронов имеют ненулевые средние значения.
- Существует корреляция между элементами вектора входного сигнала, а также между индуцированными выходными сигналами нейронов.
- При переходе от одного слоя к следующему возможны значительные вариации производных второго порядка функции стоимости по синаптическим весам нейронов сети. Вторые производные часто оказываются меньше для первых слоев, поэтому синаптические веса первых скрытых слоев обучаются медленнее, чем веса последних скрытых слоев.

Из главы 3 известно, что *время обучения* (learning time) алгоритма LMS зависит от соотношения $\lambda_{max}/\lambda_{min}$, где λ_{max} и λ_{min} — наибольшее и наименьшее ненулевое собственное число Гессиана соответственно. Экспериментальные результаты показали, что аналогичная ситуация сохраняется и для алгоритма обратного распространения, который является обобщением алгоритма LMS. Для входных сигналов с ненулевым средним значением отношение $\lambda_{max}/\lambda_{min}$ больше, чем для соответствующих значений с нулевым средним. Чем больше среднее значение входного сигнала, тем больше это отношение (см. упражнение 3.10). Это в значительной степени определяет динамику алгоритма обратного распространения.

Для минимизации времени обучения следует избегать использования входных значений с ненулевым средним. Для вектора $\mathbf{x}$, передаваемого нейронам первого скрытого слоя многослойного персептрона (т.е. вектора сигнала, поступающего на входной слой), это условие выполнить очень легко. Достаточно вычесть из каждого элемен-

[7] В [88] представлен обзор точных алгоритмов вычисления матрицы Гессиана и алгоритмов его аппроксимации с учетом специфики нейронных сетей. В этом отношении также представляет интерес [103].

та этого вектора его ненулевое среднее значение до передачи этого вектора в сеть. Однако что делать с сигналами, передаваемыми на следующие скрытые и выходной слои сети? Ответ на этот вопрос заключается в выборе типа функций активации, которые используются в сети. Если функция активации является несимметричной (как в случае логистической функции), выходной сигнал любого нейрона принадлежит интервалу $[0, 1]$. Такой выбор является источником *систематического смещения* (systematic bias) для нейронов, расположенных правее первого скрытого слоя сети. Чтобы обойти эту проблему, следует использовать антисимметричную функцию активации, такую как гиперболический тангенс. В последнем случае выход каждого нейрона может принимать как положительные, так и отрицательные значения из интервала $[-1, 1]$ и с более высокой вероятностью будет иметь нулевое среднее. Если связность сети велика, обучение методом обратного распространения с антисимметричными функциями активации приведет к более быстрой сходимости, чем в случае применения симметричной функции активации, что также является эмпирически очевидным [625]. Это объясняет эвристику 3 из раздела 4.6.

4.12. Обобщение

При обучении методом обратного распространения в сеть подают обучающую выборку и вычисляют синаптические веса многослойного персептрона, загружая в сеть максимально возможное количество примеров. При этом разработчик надеется, что обученная таким образом сеть будет способна к обобщению. Считается, что сеть обладает *хорошей обобщающей способностью* (generalize well), если отображение входа на выход, осуществляемое ею, является корректным (или близким к этому) для данных, никогда ранее не "виденных" сетью в процессе обучения. Термин "обобщение" взят из области психологии. При этом считается, что данные принадлежат той же совокупности (population), из которой они брались для обучения.

Процесс обучения нейронной сети можно рассматривать как задачу *аппроксимации кривой* (curve-fitting). Сама сеть при этом выступает как один нелинейный оператор. Такая точка зрения позволяет считать обобщение не мистическим свойством нейронной сети, а результатом хорошей нелинейной интерполяции входных данных [1146]. Сеть осуществляет корректную интерполяцию в основном за счет того, что непрерывность отдельных функций активации многослойного персептрона обеспечивает непрерывность общей выходной функции.

На рис. 4.19, *а* показано, как происходит обобщение в гипотетической сети. Нелинейное отображение входа на выход, показанное на этом рисунке, определяется сетью в результате обучения по дискретным точкам (обучающим данным). Точку, полученную в процессе обобщения и обозначенную незаштрихованным кружочком, можно рассматривать как результат выполняемой сетью интерполяции.

Нейронная сеть, спроектированная с учетом хорошего обобщения, будет осуществлять корректное отображение входа на выход даже тогда, когда входной сигнал слегка отличается от примеров, использованных для обучения сети (что и показано на рисунке). Однако, если сеть обучается на слишком большом количестве примеров, все может закончиться только запоминанием данных обучения. Это может произойти за счет нахождения таких признаков (например, благодаря шуму), которые присутствуют в примерах обучения, но не свойственны самой моделируемой функции отображения. Такое явление называют *избыточным обучением*, или *переобучением* (overtraining). Если сеть "переучена", она теряет свою способность к обобщению на аналогичных входных сигналах.

Обычно загрузка данных в многослойный персептрон таким способом требует использования большего количества скрытых нейронов, чем действительно необходимо. Это выливается в нежелательное изменение входного пространства из-за шума, который содержится в синаптических весах сети. Пример плохого обобщения вследствие простого *запоминания* (memorizing) обучающих образов показан на рис. 4.19, *б* для тех же данных, которые показаны на рис. 4.19, *а*. Результат запоминания, в сущности, представляет собой справочную таблицу – список пар "вход-выход", вычисленных нейронной сетью. При этом отображение теряет свою гладкость. Как указывается в [847], гладкость отображения входа на выход непосредственно связана с критерием моделирования, который получил название *бритвы Оккама* (Occam's razor). Сущность этого критерия состоит в выборе простейшей функции при отсутствии каких-либо дополнительных априорных знаний. В контексте предыдущего обсуждения "простейшей" является самая гладкая из функций, аппроксимирующих отображение для данного критерия ошибки, так как такой подход требует минимальных вычислительных ресурсов. Гладкость свойственна многим приложениям и зависит от масштаба изучаемого явления. Таким образом, для *плохо обусловленных* (ill-posed) отношений важно искать гладкое нелинейное отображение. При этом сеть будет способна *корректно* классифицировать новые сигналы относительно примеров обучения [1146].

Достаточный объем примеров обучения для корректного обобщения

Способность к обобщению определяется тремя факторами: размером обучающего множества и его представительностью, архитектурой нейронной сети и физической сложностью рассматриваемой задачи. Естественно, последний фактор выходит за пределы нашего влияния. В контексте остальных факторов вопрос обобщения можно рассматривать с двух различных точек зрения [495].

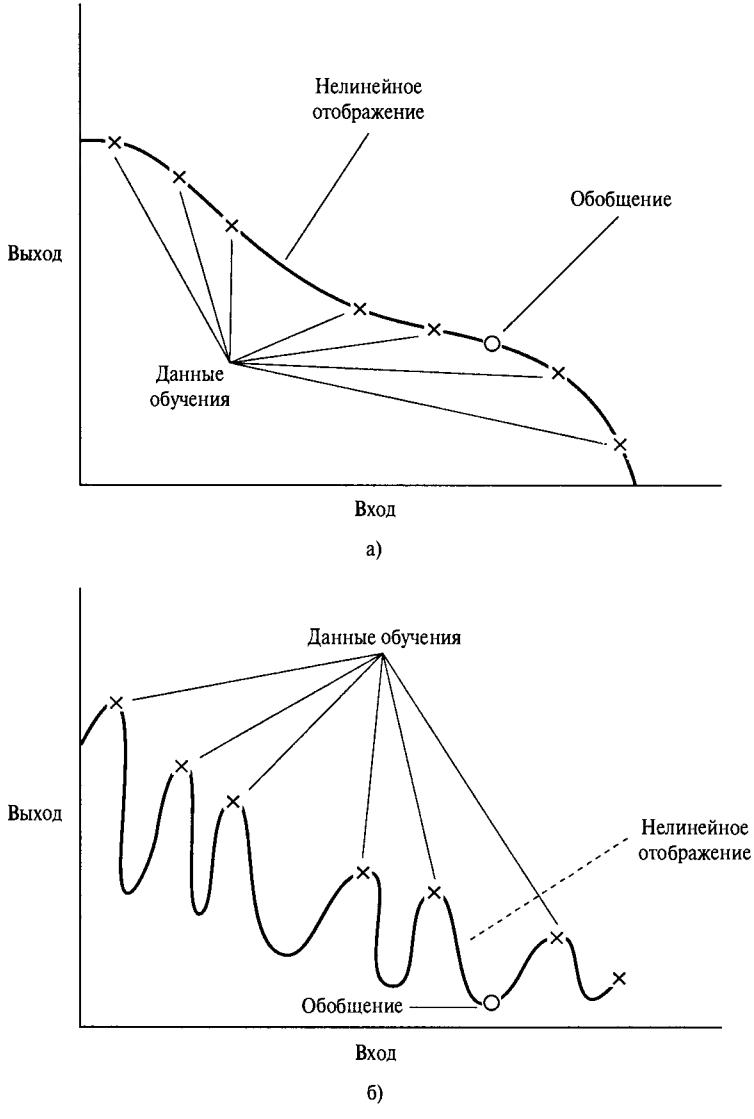


Рис. 4.19. Корректная интерполяция (хорошее обобщение) (а) и результат избыточного обучения (плохое обобщение) (б)

- Архитектура сети фиксирована (будем надеяться, в соответствии с физической сложностью рассматриваемой задачи), и вопрос сводится к определению размера обучающего множества, *необходимого* для хорошего обобщения.
- Размер обучающего множества фиксирован, и вопрос сводится к определению наилучшей архитектуры сети, позволяющей достичь хорошего обобщения.

Обе точки зрения по-своему правильны. До сих пор мы фокусировали внимание на первом аспекте проблемы.

Вопрос адекватности размера обучающей выборки рассматривался в главе 2. Там указывалось, что VC-измерение обеспечивает теоретический базис для принципиального решения этой важной задачи. В частности, были получены *независимые от распределения пессимистические* (distribution-free, worst-case) формулы оценки размера обучающего множества, *достаточного* для хорошего обобщения (см. раздел 2.14). К сожалению, часто оказывается, что между действительно достаточным размером множества обучения и этими оценками может существовать большой разрыв. Из-за этого расхождения возникает *задача сложности выборки* (sample complexity problem), открывающая новую область исследований.

На практике оказывается, что для хорошего обобщения достаточно, чтобы размер обучающего множества N удовлетворял следующему соотношению:

$$N = O(W/\varepsilon), \tag{4.85}$$

где W — общее количество свободных параметров (т.е. синаптических весов и порогов) сети; ε — допустимая точность ошибки классификации; $O(\cdot)$ — порядок заключенной в скобки величины. Например, для ошибки в 10% количество примеров обучения должно в 10 раз превосходить количество свободных параметров сети.

Выражение (4.85) получено из *эмпирического правила Видроу* (Widrow's rule of thumb) для алгоритма LMS, утверждающего, что время стабилизации процесса линейной адаптивной временной фильтрации примерно равно *объему памяти* (memory span) линейного адаптивного фильтра в задаче фильтра на *линии задержки с отводами* (tapped-delay-line filter), деленному на величину *рассогласования* (misadjustment) [1144]. Рассогласование в алгоритме LMS выступает в роли ошибки ε из выражения (4.85). Дальнейшие выкладки относительно этого эмпирического правила приводятся в следующем разделе.

4.13. Аппроксимация функций

Многослойный персептрон, обучаемый согласно алгоритму обратного распространения, можно рассматривать как практический механизм реализации *нелинейного отображения "вход-выход"* (nonlinear input-output mapping) общего вида. Например, пусть m_0 — количество входных узлов многослойного персептрона, $M = m_L$ — количество нейронов выходного слоя сети. Отношение "вход-выход" для такой сети определяет отображение m_0-мерного Евклидова пространства входных данных в M-мерное Евклидово пространство выходных сигналов, непрерывно дифференцируемое бесконечное число раз (если этому условию удовлетворяют и функции активации). При рассмотрении свойств многослойного персептрона с точки зрения отображения "вход-выход" возникает следующий фундаментальный вопрос.

Каково минимальное количество скрытых слоев многослойного персептрона, обеспечивающего аппроксимацию некоторого непрерывного отображения?

Теорема об универсальной аппроксимации

Ответ на этот вопрос обеспечивает *теорема об универсальной аппроксимации*[8] (universal approximation theorem) для нелинейного отображения "вход-выход", которая формулируется следующим образом.

Пусть $\varphi(\cdot)$ — ограниченная, не постоянная монотонно возрастающая непрерывная функция. Пусть I_{m_0} — m_0-мерный единичный гиперкуб $[0, 1]^{m_0}$. Пусть пространство непрерывных на I_{m_0} функций обозначается символом $C(I_{m_0})$. Тогда для любой функции $fC(I_{m_0})$ и $\varepsilon > 0$ существует такое целое число m_1 и множество действительных констант α_i, b_i и w_{ij}, где $i = 1, \ldots, m_1$, $j = 1, \ldots, m_0$, что

$$F(x_1, \ldots, x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi \left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i \right) \tag{4.86}$$

является реализацией аппроксимации функции $f(\cdot)$, т.е.

$$|F(x_1, \ldots, x_{m_0}) - f(x_1, \ldots, x_{m_0})| < \varepsilon$$

для всех $x_1, x_2, \ldots, x_{m_0}$, принадлежащих входному пространству.

Теорема об универсальной аппроксимации непосредственно применима к многослойному персептрону. Во-первых, заметим, что в модели многослойного персептрона в качестве функции активации используется ограниченная, монотонно возрастающая логистическая функция $1/[1+\exp(-v)]$, удовлетворяющая условиям, накладываемым теоремой на функцию $\varphi(\cdot)$. Во-вторых, заметим, что выражение (4.86) описывает выходной сигнал персептрона следующего вида.

---

[8] Универсальную теорему аппроксимации можно рассматривать как естественное расширение теоремы Вейерштрасса [1124]. Эта теорема утверждает, что любая непрерывная функция на замкнутом интервале действительной оси может быть представлена абсолютно и равномерно сходящимся рядом полиномов.

Интерес к исследованию многослойных персептронов в качестве механизма представления произвольных непрерывных функций впервые был проявлен в [448], в которой использовалась усовершенствованная теорема Колмогорова (Kolmogorov) о суперпозиции [1014]. В [333] показано, что многослойный персептрон с одним скрытым слоем, косинусоидальной пороговой функцией и линейным выходным слоем представляет собой частный случай "сети Фурье", обеспечивающей на выходе аппроксимацию заданной функции рядом Фурье. Однако в контексте обычного многослойного персептрона Цыбенко (Cybenko) впервые строго продемонстрировал, что одного скрытого слоя достаточно для аппроксимации произвольной непрерывной функции, заданной на единичном гиперкубе. Эта работа была опубликована в Техническом отчете университета штата Иллинойс в 1988 году и вышла отдельным изданием год спустя [236], [237]. В 1989 году независимо друг от друга были опубликованы две работы по использованию персептронов в качестве универсальных аппроксиматоров [329], [486]. Другие вопросы аппроксимации описаны в [642].

1. Сеть содержит m_0 входных узлов и один скрытый слой, состоящий из m_1 нейронов. Входы обозначены $x_1, x_2, \ldots, x_{m_0}$.
2. Скрытый нейрон i имеет синаптические веса $w_{i_1}, \ldots, w_{m_0}$ и порог b_i.
3. Выход сети представляет собой линейную комбинацию выходных сигналов скрытых нейронов, взвешенных синаптическими весами выходного нейрона — $\alpha_1, \ldots, \alpha_{m_1}$.

Теорема об универсальной аппроксимации является *теоремой существования* (existence theorem), т.е. математическим доказательством возможности аппроксимации любой непрерывной функции. Выражение (4.86), составляющее стержень теоремы, просто обобщает описание аппроксимации функции конечным рядом Фурье. Таким образом, теорема утверждает, что *многослойного персептрона с одним скрытым слоем достаточно для построения равномерной аппроксимации с точностью* ε *для любого обучающего множества, представленного набором входов* $x_1, x_2, \ldots, x_{m_0}$ *и желаемых откликов* $f(x_1, x_2, \ldots, x_{m_0})$. Тем не менее из теоремы не следует, что один скрытый слой является *оптимальным* в смысле времени обучения, простоты реализации и, что более важно, качества обобщения.

Пределы ошибок аппроксимации

В [96] были исследованы аппроксимирующие свойства многослойного персептрона для случая одного скрытого слоя с сигмоидальной функцией активации и одного линейного выходного нейрона. Эта сеть обучалась с помощью алгоритма обратного распространения, после чего тестировалась на новых данных. Во время обучения сети предъявлялись выбранные точки аппроксимируемой функции f, в результате чего была получена аппроксимирующая функция F, определяемая выражением (4.86). Если сети предъявлялись не использованные ранее данные, то функция F "оценивала" новые точки целевой функции, т.е. $F = \hat{f}$.

Гладкость целевой функции f выражалась в терминах ее разложения Фурье. В частности, в качестве предельной амплитуды функции f использовалось среднее значение нормы вектора частоты, взвешенного значениями амплитуды распределения Фурье. Пусть $\tilde{f}(\omega)$ — многомерное преобразование Фурье функции $f(\mathbf{x})$, $\mathbf{x} \in \Re^{m_0}$, где ω — вектор частоты. Функция $f(x)$, представленная в терминах преобразования Фурье $\tilde{f}(\omega)$, определяется следующей инверсной формулой:

$$f(x) = \int_{\Re^{m_0}} \tilde{f}(\omega) \exp(j\omega^T \mathbf{x}) d\omega, \tag{4.87}$$

где $j = \sqrt{-1}$. Для комплекснозначной функции $\tilde{f}(\omega)$ с интегрируемой функцией $\omega\tilde{f}(\omega)$ *первый абсолютный момент* (first absolute moment) *распределения Фурье* (Fourier magnitude distribution) функции f можно определить следующим образом:

$$C_f = \int_{\Re^{m_0}} |\tilde{f}(\omega)| \times \|\omega\|^{1/2} d\omega, \tag{4.88}$$

где $||\omega||$ – Евклидова норма вектора ω; $|\tilde{f}(\omega)|$ – абсолютное значение функции $\tilde{f}(\mathbf{w})$. Первый абсолютный момент C_f является мерой *гладкости* (smoothness) функции f.

Первый абсолютный момент C_f является основой для вычисления пределов ошибки, которая возникает вследствие использования многослойного персептрона, представленного функцией отображения "вход-выход" $F(\mathbf{x})$, аппроксимирующей функцию $f(\mathbf{x})$. Ошибка аппроксимации измеряется *интегральной квадратичной ошибкой* (integrated squared error) по произвольной мере вероятности μ для шара $B_r = \{\mathbf{x}: ||\mathbf{x}|| \leq r\}$ радиуса $r > 0$. На этом основании можно сформулировать следующее утверждение для предела ошибки аппроксимации [96].

Для любой непрерывной функции $f(\boldsymbol{x})$ с конечным первым моментом C_f и любого $m_1 \geq 1$ существует некоторая линейная комбинация сигмоидальных функций $F(\boldsymbol{x})$ вида (4.86), такая, что

$$\int_{B_r} (f(\boldsymbol{x}) - F(\boldsymbol{x}))^2 \mu(d\boldsymbol{x}) \leq \frac{C_f'}{m_1},$$

где $C_f' = (2rC_f)^2$.

Если функция $f(\mathbf{x})$ наблюдается на множестве значений $\{\mathbf{x}_i\}_{i=1}^N$ входного вектора $\mathbf{x}$, принадлежащего шару B_r, этот результат определяет следующее ограничение для *эмпирического риска* (empirical risk):

$$R = \frac{1}{N}\sum_{i=1}^{N} (f(\boldsymbol{x}_i) - F(\boldsymbol{x}_i))^2 \leq \frac{C_f'}{m_1}. \tag{4.89}$$

В [95] результат (4.89) использовался для описания границ риска R, возникающего при использовании многослойного персептрона с m_0 входными узлами и m_1 скрытыми нейронами:

$$R \leq O\left(\frac{C_f^2}{m_1}\right) + O\left(\frac{m_0 m_1}{N}\log N\right). \tag{4.90}$$

Два слагаемых в этом определении границ риска R отражают компромисс между двумя противоречивыми требованиями к размеру скрытого слоя.

1. *Точность наилучшей аппроксимации* (accuracy of the best approximation). В соответствии с теоремой об универсальной аппроксимации для удовлетворения этого требования размер скрытого слоя m_1 должен быть большим.
2. *Точность эмпирического соответствия аппроксимации* (accuracy of empirical fit to the approximation). Для того чтобы удовлетворить этому требованию, отношение m_1/N должно иметь малое значение. Для фиксированного объема N обучающего множества размер скрытого слоя m_1 должен оставаться малым, что противоречит первому требованию.

Ограничение для риска R, описанное формулой (4.90), имеет еще одно интересное применение. Дело в том, что для точной оценки целевой функции *не* требуется экспоненциально большого обучающего множества и большой размерности входного пространства m_0, если первый абсолютный момент C_f остается конечным. Это еще больше повышает практическую ценность многослойного персептрона, используемого в качестве универсального аппроксиматора.

Ошибку между *эмпирическим соответствием* (empirical fit) и наилучшей аппроксимацией можно рассматривать как *ошибку оценивания* (estimation error), описанную в главе 2. Пусть ε_0 — среднеквадратическое значение ошибки оценивания. Тогда, игнорируя логарифмический множитель во втором слагаемом неравенства (4.90), можно сделать вывод, что размер N обучающего множества, необходимый для хорошего обобщения, должен иметь порядок $m_0 m_1/\varepsilon_0$. Математическая структура этого результата аналогична эмпирическому правилу (4.85), если произведение $m_0 m_1$ соответствует общему количеству свободных параметров W сети. Другими словами, можно утверждать, что для хорошего качества аппроксимации размер обучающего множества должен превышать отношение общего количества свободных параметров сети к среднеквадратическому значению ошибки оценивания.

"Проклятие размерности"

Из ограничения (4.90) вытекает еще один интересный результат. Если размер скрытого слоя выбирается по следующей формуле (т.е. риск R минимизируется по N):

$$m_1 = C_f \left(\frac{N}{m_0 \log N} \right)^{1/2},$$

то риск R ограничивается величиной $O(C_f \sqrt{m_0 (\log N/N)})$. Неожиданный аспект этого результата состоит в том, что в терминах поведения риска R скорость сходимости, представленная как функция от размера обучающего множества N, имеет порядок $(1/N)^{1/2}$ (умноженный на логарифмический член). В то же время обычная гладкая функция (например, тригонометрическая или полиномиальная) демонстрирует несколько другое поведение. Пусть s — мера гладкости, определяемая как степень

дифференцируемости функции (количество существующих производных). Тогда для обычной гладкой функции минимаксная скорость сходимости общего риска R имеет порядок $(1/N)^{2s/(2s+m_0)}$. Зависимость этой скорости от размерности входного пространства m_0 называют "*проклятием размерности*" (curse of dimensionality), поскольку это свойство ограничивает практическое использование таких функций. Таким образом, использование многослойного персептрона для решения задач аппроксимации обеспечивает определенные преимущества перед обычными гладкими функциями. Однако это преимущество появляется при условии, что первый абсолютный момент C_f остается конечным. В этом состоит ограничение гладкости.

Термин "проклятие размерности" (curse of dimensionality) был введен Ричардом Белманом (Richard Belman) в 1961 году в работе, посвященной процессам адаптивного управления [117]. Для геометрической интерпретации этого понятия рассмотрим пример, в котором $\mathbf{x}$ — m_0-мерный входной вектор, а множество $\{(\mathbf{x}_i, d_i)\}, i = 1, 2, \ldots, N$, задает обучающую выборку. *Плотность дискретизации* (sampling density) пропорциональна значению N^{1/m_0}. Пусть $f(\mathbf{x})$ — поверхность в m_0-мерном входном пространстве, проходящая около точек данных $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$. Если функция $f(\mathbf{x})$ достаточно сложна и (по большей части) абсолютно неизвестна, необходимо *уплотнить* (dense) точки данных для более полного изучения поверхности. К сожалению, в многомерном пространстве из-за "проклятия размерности" очень сложно найти обучающую выборку с высокой плотностью дискретизации. В частности, в результате увеличения размерности наблюдается *экспоненциальный* рост сложности, что, в свою очередь, приводит к ухудшению пространственных свойств случайных точек с равномерным распределением. Основная причина "проклятия размерности" обосновывается в [321].

> *Функция, определенная в пространстве большой размерности, скорее всего, является значительно более сложной, чем функция, определенная в пространстве меньшей размерности, и эту сложность трудно разглядеть.*

Единственной возможностью избежать "проклятия размерности" является получение корректных *априорных знаний* о функции, определяемой данными обучения.

Можно утверждать, что для практического получения хорошей оценки в пространствах высокой размерности необходимо обеспечить возрастание гладкости неизвестной функции наряду с увеличением размерности входных данных [787]. Эта точка зрения будет обоснована в главе 5.

Практические соображения

Теорема об универсальной аппроксимации является очень важной с теоретической точки зрения, так как она обеспечивает *необходимый математический базис* для доказательства применимости сетей прямого распространения с одним скрытым слоем для решения задач аппроксимации. Без такой теоремы можно было бы безрезуль-

татно заниматься поисками решения, которого на самом деле не существует. Однако эта теорема не конструктивна, поскольку она не обеспечивает способ нахождения многослойного персептрона, обладающего заявленными свойствами аппроксимации.

Теорема об универсальной аппроксимации предполагает, что аппроксимируемая непрерывная функция известна, и для ее приближения можно использовать скрытый слой неограниченного размера. В большинстве практических применений многослойного персептрона оба эти предположения нарушаются.

Проблема многослойного персептрона с одним скрытым слоем состоит в том, что нейроны могут взаимодействовать друг с другом на глобальном уровне. В сложных задачах такое взаимодействие усложняет задачу повышения качества аппроксимации в одной точке без явного ухудшения в другой. С другой стороны, при наличии двух скрытых слоев процесс аппроксимации становится более управляемым. В частности, можно утверждать следующее [190], [329].

1. *Локальные признаки* (local feature) извлекаются в первом скрытом слое, т.е. некоторые скрытые нейроны первого слоя можно использовать для разделения входного пространства на отдельные области, а остальные нейроны слоя обучать локальным признакам, характеризующим эти области.
2. *Глобальные признаки* (global feature) извлекаются во втором скрытом слое. В частности, нейрон второго скрытого слоя "обобщает" выходные сигналы нейронов первого скрытого слоя, относящихся к конкретной области входного пространства. Таким образом он обучается глобальным признакам этой области, а в остальных областях его выходной сигнал равен нулю.

Этот двухэтапный процесс аппроксимации по своей философии аналогичен сплайновому подходу к аппроксимации кривых, поскольку нейроны работают в изолированных областях. *Сплайн* (spline) является примером такой кусочной полиномиальной аппроксимации.

В [1007] предложено дальнейшее обоснование использования двух скрытых слоев в контексте *обратных задач* (inverse problem). В частности, рассматривается следующая обратная задача.

Для данной непрерывной вектор-функции $\mathbf{f} : \Re^m \rightarrow \Re^M$, *компактного подмножества* $\mathbf{C} \subseteq \Re^M$, *которое содержится в пространстве образов функции* $\mathbf{f}$, *и некоторого положительного* $\varepsilon > 0$ *требуется найти вектор-функцию* $\varphi : \Re^M \rightarrow \Re^m$, *удовлетворяющую условию*

$$||\varphi(\mathbf{f})(\mathbf{u}) - \mathbf{u}|| < \varepsilon \text{ для любого } \mathbf{u} \in \mathbf{C}.$$

Эта задача относится к области *обратной кинематики* (inverse kinematics) или динамики, где наблюдаемое состояние $\mathbf{x}(n)$ системы является функцией текущих действий $\mathbf{u}(n)$ и предыдущего состояния $\mathbf{x}(n-1)$ системы

$$\mathbf{x}(n) = \mathbf{f}(\mathbf{x}(n-1), \mathbf{u}(n)).$$

Здесь предполагается, что функция $\mathbf{f}$ является *обратимой* (invertible), т.е. $\mathbf{u}(n)$ можно представить как функцию от $\mathbf{x}(n)$ для любого $\mathbf{x}(n-1)$. Функция $\mathbf{f}$ описывает прямую кинематику, а функция φ — обратную. В контексте излагаемого материала необходимо построить такую функцию φ, которая может быть реализована многослойным персептроном. В общем случае для решения обратной задачи кинематики функция φ должна быть разрывной. Интересно, что для решения таких обратных задачи одного скрытого слоя *недостаточно*, даже при использовании нейронной модели с разрывными активационными функциями, а персептрона с двумя скрытыми слоями вполне достаточно для любых возможных $\mathbf{C}$, $\mathbf{f}$ и ε [1007].

4.14. Перекрестная проверка

Сущность обучения методом обратного распространения заключается в кодировании отображения входа на выход (представленного множеством маркированных примеров) в синаптических весах и пороговых значениях многослойного персептрона. Предполагается, что на примерах из прошлого сеть будет обучена настолько хорошо, что сможет обобщить их на будущее. С такой точки зрения процесс обучения обеспечивает настройку параметров сети для заданного множества данных. Более того, проблему настройки сети можно рассматривать как задачу выбора наилучшей модели из множества структур-"кандидатов" с учетом определенного критерия.

Основой для решения такой задачи может стать стандартный статистический подход, получивший название *перекрестной проверки* (cross-validation)[9] [1019], [1020]. В рамках этого подхода имеющиеся в наличии данные сначала случайным образом разбиваются на *обучающее множество* (training set) и *тестовое множество* (test set). Обучающее множество, в свою очередь, разбивается на два следующих несвязанных подмножества.

- *Подмножество для оценивания* (estimation subset), используемое для выбора модели.
- *Проверочное подмножество* (validation subset), используемое для тестирования модели.

[9] История развития методологии "перекрестной проверки" описана в [1020]. Сама идея перекрестных проверок витала в воздухе еще в 1930-е годы, однако в виде технологии она оформилась лишь в 1960–1970-х годах. Важный вклад в развитие этого подхода внесли Стоун [1020] и Гессер [341], которые одновременно и независимо друг от друга представили эту методологию. Стоун назвал ее "методом перекрестной проверки", а Гессер – "прогнозируемым методом повторного использования обучающей выборки" (predictive sample reuse method).

Идея этого подхода состоит в проверке качества модели на данных, отличных от использованных для параметрического оценивания. Таким образом, обучающее множество можно использовать для проверки эффективности различных моделей-"кандидатов", из которых необходимо выбрать лучшую. Однако существует некоторая вероятность того, что отобранная по параметрам эффективности модель окажется излишне *переученной* (overfitted) на проверочном подмножестве. Чтобы предотвратить такую опасность, эффективность обобщения выбранной модели измеряется также на тестовом множестве, которое отличается от проверочного.

Перекрестные проверки особенно привлекательны, если требуется создать большую нейросеть, обладающую хорошей способностью к обобщению. Например, перекрестную проверку можно использовать для нахождения многослойного персептрона с оптимальным количеством скрытых нейронов. Способ определения момента остановки процесса обучения описывается в следующих разделах.

Выбор модели

Идея выбора модели на основе перекрестной проверки аналогична методологии минимизации структурного риска (см. главу 2). Рассмотрим следующую вложенную структуру классов булевых функций:

$$\begin{aligned} &\mathbf{F}_1 \subset \mathbf{F}_2 \subset \ldots \subset \mathbf{F}_n, \\ &\mathbf{F}_k = \{F_k\} = \{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathbf{W}_k\}, k = 1, 2, \ldots, n. \end{aligned} \tag{4.91}$$

Другими словами, k-й класс функций $\mathbf{F}_k$ охватывает семейство многослойных персептронов с одинаковой архитектурой и вектором весов $\mathbf{w}$, принадлежащим многомерному пространству весовых коэффициентов $\mathbf{W}_k$. Функция-член этого класса, характеризуемая функцией (или гипотезой) $F_k = F(\mathbf{w}, \mathbf{x})$, $\mathbf{w} \in \mathbf{W}_k$, отображает входной вектор $\mathbf{x}$ в множество $\{0, 1\}$, где $\mathbf{x}$ принадлежит входному пространству $\mathbf{X}$ с некоторой неизвестной вероятностью P. Каждый многослойный персептрон в описанной структуре обучается по алгоритму обратного распространения, в соответствии с которым настраиваются параметры персептрона. Задача выбора модели состоит в выборе персептрона, имеющего наилучшее значение W, определяющее количество свободных параметров (т.е. синаптических весов и порогов). Более точно, зная желаемый скалярный отклик $d = \{0, 1\}$ для входного сигнала $\boldsymbol{x}$, мы определим ошибку обобщения:

$$\varepsilon_g(F) = P(F(\mathbf{x}) \neq d) \text{ для } \mathbf{x} \in \mathbf{X}.$$

Пусть имеется множество маркированных примеров обучения

$$\mathbf{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N}.$$

Требуется выбрать такую гипотезу $F(\mathbf{x}, \mathbf{w})$, которая минимизирует ошибку обобщения $\varepsilon_g(F)$ на всем множестве примеров обучения.

В дальнейшем будем предполагать, что описанная соотношением (4.91) структура обладает следующим свойством: для любого размера обучающей выборки N всегда найдется многослойный персептрон с достаточно большим количеством свободных параметров $W_{\max}(N)$, адекватно представляющий обучающее множество $\mathbf{T}$. Это всего лишь еще одна формулировка теоремы об универсальной аппроксимации из раздела 4.13. Будем называть $W_{\max}(N)$ *числом сглаживания* (fitting number). Значение величины $W_{\max}(N)$ состоит в том, что разумная процедура выбора модели должна найти такую гипотезу $F(\mathbf{x}, \mathbf{w})$, которая удовлетворяет условию $W \leq W_{\max}(N)$. В противном случае сложность сети должна увеличиваться.

Пусть параметр r, принадлежащий интервалу от нуля до единицы, определяет разбиение обучающего множества на подмножества оценивания и проверки. Пусть $\mathbf{T}$ содержит N примеров. Тогда после разбиения $(1 - r)N$ примеров будет принадлежать подмножеству оценивания, а rN примеров — проверочному подмножеству. Подмножество оценивания обозначим $\mathbf{T}'$. Оно используется для обучения вложенной последовательности многослойных персептронов, представленных гипотезами $\mathbf{F}_1$, $\mathbf{F}_2, \ldots, \mathbf{F}_n$ с нарастающей сложностью. Так как подмножество $\mathbf{T}'$ состоит из $(1-r)N$ примеров, будем рассматривать значения W, не превышающие соответствующего числа сглаживания $W_{\max}((1 - r)N)$.

Используя результаты перекрестной проверки при выборе

$$\mathbf{F}_{cv} = \min_{k=1,2,\ldots,v} \{e''_t(\mathbf{F}_k)\}, \tag{4.92}$$

где v соответствует $W_v \leq W_{\max}((1 - r)N)$, а $e_t''(\mathbf{F}_k)$ — ошибка классификации, обеспечиваемая гипотезой $\mathbf{F}_k$ при тестировании на проверочном множестве $\mathbf{T}''$, состоящем из N примеров.

Ключевой вопрос состоит в том, как определить значение параметра r, задающего разбиение обучающего множества $\mathbf{T}$ на подмножества оценивания $\mathbf{T}'$ и проверки $\mathbf{T}''$. В [549] при аналитическом исследовании этого вопроса с использованием VC-измерения и компьютерного моделирования были определены некоторые количественные свойства оптимального значения r.

- Если сложность целевой функции, определяющей желаемый отклик d в терминах входного вектора $\mathbf{x}$, мала по сравнению с размером N обучающего множества, то эффективность перекрестной проверки мало зависит от выбора r.
- Если же целевая функция становится более сложной по сравнению с размером N обучающего множества, то выбор оптимального значения r оказывает существенное влияние на эффективность перекрестной проверки, причем это значение уменьшается.

- Одно и то же *фиксированное* значение *r почти оптимально* подходит для широкого диапазона сложностей целевой функции.

На основе результатов, полученных в [549], было получено фиксированное значение r =0,2, при котором 80% обучающего множества **T** составляет подмножество оценивания, а остальные 20% — проверочное подмножество.

Ранее речь шла о последовательности вложенных многослойных персептронов возрастающей сложности. Для заданной размерности входного и выходного слоев такую последовательность из $v = p + q$ полносвязных многослойных персептронов можно создать, например, следующим образом.

- Сформировать p многослойных персептронов с одним скрытым слоем возрастающего размера $h'_1 < h'_2 < \ldots < h'_p$.
- Сгенерировать q многослойных персептронов с двумя скрытыми слоями, размер первого скрытого слоя которых фиксирован и составляет h'_p, а размер второго — возрастает $h''_1 < h''_2 < \ldots < h''_q$.

По мере перехода от одного многослойного персептрона к следующему увеличивается соответствующее количество свободных параметров W. Процедура выбора модели, основанная на перекрестной проверке, обеспечивает принципиальный подход к определению нужного количества скрытых нейронов многослойного персептрона. Несмотря на то что эта процедура была описана в контексте двоичной классификации, она применима и к другим приложениям многослойного персептрона.

Метод обучения с ранним остановом

Обычно обучение многослойного персептрона методом обратного распространения происходит поэтапно — переходя от более простых к более сложным функциям отображения. Это объясняется тем фактом, что при нормальных условиях среднеквадратическая ошибка уменьшается по мере увеличения количества эпох обучения: она начинается с довольно больших значений, стремительно уменьшается, а затем продолжает убывать все медленнее по мере продвижения сети к локальному минимуму на поверхности ошибок. Если главной целью является хорошая способность к обобщению, то по виду кривой довольно сложно определить момент, когда следует остановить процесс обучения. В частности, как следует из материала, изложенного в разделе 4.12, если вовремя не остановить сеанс обучения, то существенно повышается вероятность излишнего *переобучения* (overfitting).

Наступление стадии излишнего переобучения можно определить с помощью перекрестной проверки, в которой данные разбиты на два подмножества — оценивания и проверки. Множество оценивания используется для обычного обучения сети с небольшой модификацией: сеанс обучения периодически останавливается (через

каждые несколько эпох), после чего сеть тестируется на проверочном подмножестве. Более точно, периодический процесс оценивания-тестирования выполняется следующим образом.

- После завершения этапа оценивания (обучения) синаптические веса и уровни порогов многослойного персептрона фиксируются, и сеть переключается в режим прямого прохода. Ошибка сети вычисляется для каждого примера из проверочного подмножества.
- После завершения тестирования наступает следующий этап процесса обучения, и все повторяется.

Такая процедура называется *методом обучения с ранним остановом* (early stopping method of training)[10].

На рис. 4.20 показана форма двух кривых обучения. Первая относится к измерениям на подмножестве для оценивания, а вторая — к измерениям на проверочном подмножестве. Обычно на проверочном подмножестве модель ведет себя не так хорошо, как на подмножестве для оценивания, на основе которого формируется архитектура сети. *Функция обучения на подмножестве оценивания* (estimation learning curve) монотонно убывает с увеличением количества эпох, в то время как *кривая обучения на проверочном подмножестве* (validation learning curve) монотонно убывает до некоторого минимума, после чего при продолжении обучения начинает возрастать. Если учитывать только кривую, построенную на подмножестве оценивания, то возникает соблазн продолжать обучение и после прохождения точки минимума на кривой, построенной на проверочном подмножестве. Однако на практике оказывается, что после прохождения этой точки сеть обучается только посторонним шумам, содержащимся в обучающей выборке. Поэтому можно выдвинуть эвристическое предположение о том, что точку минимума кривой тестирования можно использовать в качестве критерия останова сеанса обучения.

А что если данные обучения не содержат шума? Как в этом случае определить точку раннего останова для детерминированного сценария? Частично на этот вопрос можно ответить следующим образом. Если ошибки на подмножествах оценивания и проверки не одинаково стремятся к нулю, то емкости сети не достаточно для точного функционирования модели. Лучшее, что можно сделать в таком случае, – это постараться минимизировать *интегрированную квадратичную ошибку* (integrated squared error), что (приблизительно) эквивалентно минимизации обычной глобальной среднеквадратической ошибки при равномерном распределении входного сигнала.

---

[10] Первые упоминания о методе обучения с ранним остановом содержатся в [754], [1121]. Пожалуй, наиболее полный статистический анализ этого метода обучения многослойного персептрона представлен в [40]. Это исследование подтверждается результатами компьютерного моделирования для классификатора с архитектурой 8-8-4, 108-ю настраиваемыми параметрами при чрезвычайно большом множестве данных (50000 примеров).

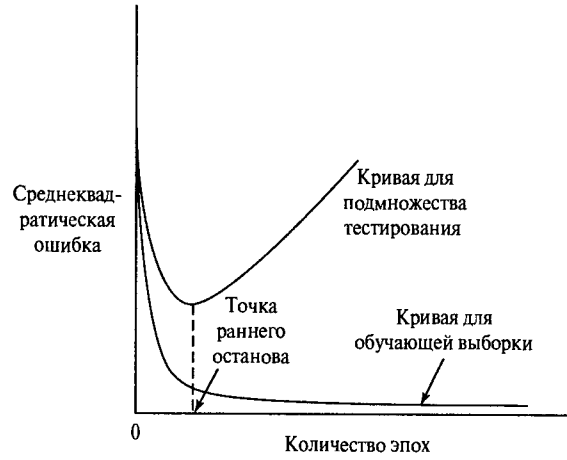


Рис. 4.20. Процесс обучения с ранним остановом на основе перекрестной проверки

Статистическая теория явления чрезмерного *переобучения* (overfitting) описана в [37]. В этой работе содержится предостережение против использования метода раннего останова процесса обучения. Данная теория основывается на пакетном обучении и подтверждается основательным компьютерным моделированием классификатора на основе многослойного персептрона с одним скрытым слоем. В зависимости от размера обучающего множества выделяются два типа поведения (режима).

Первый — это *неассимптотический режим* (nonasymptotic mode), при котором $N < 30W$, где N — размер обучающего множества; W — количество свободных параметров сети. В этом режиме метод раннего останова повышает эффективность обобщения по сравнению с полным обучением (при котором используется все множество примеров без останова сеанса). Согласно полученным результатам, переучивание может наступить при $N < 30W$, и только в этом случае существует практическая польза от использования перекрестной проверки для останова сеанса обучения. Оптимальное значение параметра r, определяющее разбиение множества примеров на подмножества для оценивания и проверки, должно выбираться из соотношения

$$r_{\text{opt}} = 1 - \frac{\sqrt{2W-1}-1}{2(W-1)}.$$

При больших значениях W эта формула сводится к следующей:

$$r_{\text{opt}} \simeq 1 - \frac{1}{\sqrt{2W}}, \quad W >> 0. \tag{4.93}$$

Например, при $W = 100$, $r = 0,07$. Это значит, что 93% данных обучения составляет подмножество оценивания и только 7% приходится на проверочное подмножество.

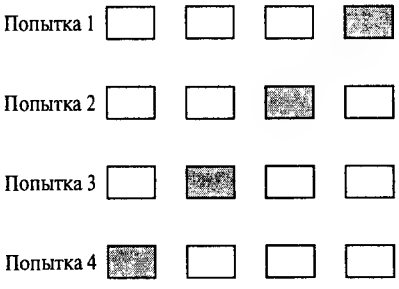


Рис. 4.21. Метод многократной перекрестной проверки. При каждой попытке обучения заштрихованное подмножество используется для вычисления ошибки оценивания

Вторым типом поведения является *асимптотический режим* (asymptotic mode), при котором $N > 30W$. В этом режиме при использовании метода раннего останова наблюдается лишь небольшое улучшение эффективности обобщения по сравнению с полным обучением. Другими словами, *полное обучение* (exhaustive learning) приводит к удовлетворительному результату, если размер множества обучения велик по сравнению с количеством параметров сети.

Варианты метода перекрестной проверки

Предложенный подход к реализации перекрестной проверки получил название *метода удержания* (hold out method). Существуют и другие варианты перекрестной проверки, которые также нашли свое применение на практике, особенно при недостатке маркированных примеров. В такой ситуации можно использовать *многократную перекрестную проверку* (multifold cross-validation), разделив имеющееся в наличие множество из N примеров на K подмножеств ($K > 1$). При этом подразумевается, что N кратно K. Обучение модели проводится на всех подмножествах, кроме одного. На этом оставшемся подмножестве выполняется тестирование и определяется ошибка оценивания. Эта процедура повторяется K раз, при этом каждый раз для тестирования используются разные подмножества (случай для $K = 4$ показан на рис. 4.21). Эффективность такой модели вычисляется путем усреднения квадратичной ошибки по всем попыткам. Метод многократной перекрестной проверки имеет один недостаток: он требует очень большого объема вычислений, так как обучение должно проводится K раз ($1 < K < N$).

Если количество доступных примеров N ограничено, можно использовать экстремальную форму многократной перекрестной проверки — метод исключения одного примера. В этом случае для обучения модели используется $N - 1$ примеров, а тестирование выполняется на оставшемся. Эксперимент повторяется N раз, причем каждый раз для проверки используются разные примеры. После этого квадратичная ошибка оценки усредняется по всем N экспериментам.

4.15. Методы упрощения структуры сети

Решение реальных задач с помощью нейронных сетей обычно требует использования четко структурированных сетей довольно большого размера. В этом контексте возникает практический вопрос минимизации размера сети без потери производительности. При уменьшении размера нейронной сети снижается вероятность обучения помехам, содержащимся в примерах, и, таким образом, повышается качество обобщения. Минимизировать размер сети можно двумя способами.

- *Наращивание сети* (network growing). В этом случае в качестве начальной архитектуры выбирается простой многослойный персептрон, возможностей которого явно недостаточно для решения рассматриваемой задачи, после чего в сеть добавляется новый нейрон или слой скрытых нейронов только в том случае, если ошибка в процессе обучения перестает уменьшаться[11].
- *Упрощение структуры сети* (network pruning). В этом случае процесс адаптации начинается с большого многослойного персептрона, производительности которого достаточно для решения поставленной задачи. После этого сеть постепенно упрощается за счет избирательного или последовательного ослабления или отключения отдельных синаптических связей.

В данном разделе мы сосредоточим внимание на втором способе — процедуре упрощения. В частности, будут описаны два подхода, первый из которых основывается на одной из форм "регуляризации", а второй — на удалении из сети определенных синаптических связей.

---

[11] Примером реализации подхода наращивания сетн *является* обучение методом каскадной корреляции (cascade-correlation learning) [288]. Процедура обучения начинается с минимальной структуры сети, имеющей несколько входных и выходных узлов, что определяется условиями задачи, и не содержащей скрытых узлов. Для обучения сети может использоваться любой алгоритм, например LMS. Скрытые нейроны добавляются в сеть по одному, образуя таким образом многослойную структуру добавленных скрытых нейронов. После добавления нового нейрона его входные сннаптические связи замораживаются, а обучаются только его выходные сиинаптическне связн. После их обучения новый скрытый нейрон становится постоянным детектором признаков данной сети. Процедура добавлення новых скрытых нейронов описанным выше способом продолжается до тех пор, пока не удастся достичь удовлетворительной производительности сети.

Еще один подход к наращиванию сети описан в [628]. В ней к прямому (функциональному) и обратному проходу (адаптации параметров) добавлен третий проход — адаптации структурного уровня (structure-level adaptation). На этом третьем уровне вычислений структура сети настраивается посредством изменения количества нейронов и структурных связей между ними. При этом используется следующий критерий: если ошибка оценивання (после определенного чнсла итераций обучения) больше некоторого заданного значения, в сеть добавляется новый нейрон. Причем он добавляется в то место, где потребность в нем максимальна. Это местоположение определяется с помощью мониторинга поведення сети прн обучении. В частности, еслн после продолжительного периода адаптации компоненты вектора синаптических весов, относящиеся к данному нейрону, продолжают сильно колебаться, значит, этому нейрону не хватает вычислительной мощности для обучения. В процессе структурной адаптации допускается также аннулирование (исключение) нейронов. Нейрон аннулируется, если он не является функциональным элементом сети нли является ее избыточным элементом. Этот метод наращивания сетн является очень ресурсоемким и требует интенсивных вычислений.

Регуляризация сложности

При создании многослойного персептрона строится нелинейная *модель* физического явления, обеспечивающая обобщение примеров типа "вход-выход", использованных при обучении сети. Поскольку архитектура сети по своей природе статична, необходимо обеспечить баланс между достоверностью данных обучения и качеством самой модели. В контексте обучения методом обратного распространения или любого другого метода обучения с учителем этого компромисса можно достичь с помощью минимизации общего риска:

$$R(\mathbf{w}) = \mathbf{E}_s(\mathbf{W}) + \lambda \mathbf{E}_c(\mathbf{w}). \tag{4.94}$$

Первое слагаемое — $\mathbf{E}_s(\mathbf{w})$ — это стандартная *мера эффективности* (performance measure), которая зависит как от самой сети (модели), так и от входных данных. При обучении методом обратного распространения она обычно определяется как среднеквадратическая ошибка, которая вычисляется по всем выходным нейронам сети на всем обучающем множестве примеров для каждой эпохи. Второе слагаемое — $\mathbf{E}_c(\mathbf{w})$ — *штраф за сложность* (complexity penalty), который зависит исключительно от самой сети (модели). Его оценка основывается на предварительных знаниях о рассматриваемой модели. На самом деле формула общего риска, определяемая соотношением (4.94), является одним из утверждений *теории регуляризации Тихонова*. Этот вопрос подробно обсуждается в главе 5. Следует подчеркнуть, что λ является *параметром регуляризации* (regularization parameter), который характеризует относительную значимость слагаемого штрафа за сложность по сравнению со слагаемым, описывающим меру производительности. Если параметр λ равен нулю, процесс обучения методом обратного распространения ничем не ограничивается и архитектура сети полностью определяется предоставленными примерами обучения. С другой стороны, если параметр λ является бесконечно большим, то ограничение, представляющее штраф за сложность, самодостаточно для определения архитектуры сети. Другими словами, в этом случае примеры обучения считаются недостоверными. В практических реализациях этой процедуры взвешивания параметру регуляризации λ назначается некоторое среднее значение. Описываемая здесь точка зрения на использование регуляризации сложности для улучшения обобщающей возможности сети вполне согласуется с процедурой минимизации структурного риска, представленной в главе 2.

В общем случае одним из вариантов выбора слагаемого штрафа за сложность является сглаживающий интеграл k-го порядка:

$$\mathbf{E}_c(\mathbf{w}, k) = \frac{1}{2} \int \left\| \frac{\partial^k}{\partial \mathbf{x}^k} F(\mathbf{x}, \mathbf{w}) \right\|^2 \mu(\mathbf{x}) d\mathbf{x}, \tag{4.95}$$

где $F(\mathbf{x}, \mathbf{w})$ — выполняемое моделью отображение входа на выход; $\mu(\mathbf{x})$ — некоторая весовая функция, определяющая область входного пространства, на которой функция $F(\mathbf{x}, \mathbf{w})$ должна быть гладкой. Это делается для того, чтобы k-я производная функции $F(\mathbf{x}, \mathbf{w})$ по входному вектору $\mathbf{x}$ принимала малое значение. Чем больше величина k, тем более гладкой (т.е. менее сложной) будет функция $F(\mathbf{x}, \mathbf{w})$.

Далее описываются три процедуры регуляризации сложности (в порядке усложнения) для многослойного персептрона.

Снижение весов

В *процедуре снижения весов* (weight decay) [458] слагаемое штрафа за сложность определяется как квадрат нормы вектора весовых коэффициентов $\mathbf{w}$ (т.е. всех свободных параметров) сети:

$$\mathbf{E}_c(\mathbf{w}) = \|\mathbf{w}\|^2 = \sum_{i \in \mathbf{C}_{\text{total}}} w_i^2, \tag{4.96}$$

где $\mathbf{C}_{\text{total}}$ — множество всех синаптических весов сети. Эта процедура обеспечивает близость к нулю отдельных синаптических весов сети, одновременно допуская достаточно большие значения остальных синаптических весов. В результате синаптические веса группируются в две категории: те, которые оказывают большое влияние на сеть (модель), и те, которые имеют малое влияние. Синаптические веса, относящиеся ко второй группе, называют *избыточными* (excess weight). Без регуляризации сложности эти веса приводят к снижению качества обобщения, так как они могут принимать совершенно произвольные значения, а также заставляют сеть для незначительного уменьшения ошибки выполнять обучение на больших объемах данных [495]. Использование процедуры регуляризации сложности приводит к уменьшению значений избыточных весов до значений, близких к нулю, что повышает качество обобщения.

В процедуре снижения весов ко всем весам многослойного персептрона применяется единый подход. Таким образом, предполагается, что центр упомянутого ранее распределения в пространстве весов находится в начале координат. Строго говоря, снижение весов не является корректной формой регуляризации сложности многослойного персептрона, так как эта процедура не вписывается в логику соотношения (4.95). Тем не менее этот алгоритм достаточно прост и неплохо работает в некоторых приложениях.

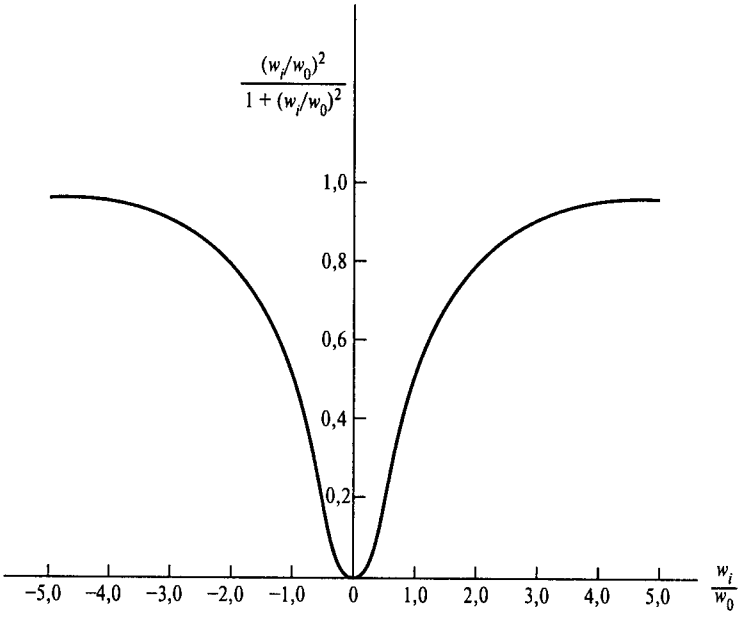


Рис. 4.22. Зависимость значения штрафа за сложность $(w_i/w_0)^2/[1+(w_i/w_0)^2]$ от w_i/w_0

Исключение весов

Во второй по сложности процедуре *исключения весов* (weight elimination) штраф за сложность определяется следующей величиной [1122]:

$$\mathbf{E}_c(\mathbf{w}) = \sum_{i \in \mathbf{C}_{\text{total}}} \frac{(w_i/w_0)^2}{1+(w_i/w_0)^2}, \tag{4.97}$$

где w_0 — некоторый предопределенный параметр; w_i — вес i-го синапса сети. Под $\mathbf{C}_{total}$ понимается множество всех синаптических связей сети. Отдельные слагаемые штрафа за сложность симметричны относительно частного w_i/w_0, что наглядно показано на рис. 4.22. Если $|w_i| >> w_0$, штраф за сложность (стоимость) для этого веса достигает максимального значения — единицы. Это значит, что вес w_i имеет высокую ценность для процесса обучения методом обратного распространения. Таким образом, слагаемое штрафа за сложность в выражении (4.97) выполняет свою задачу — позволяет выявлять синаптические веса, оказывающие первостепенное влияние на сеть. Обратите внимание, что снижение весов является частным случаем процедуры исключения весов. В частности, для больших значений w_0 выражение (4.97) сводится к (4.96) (не учитывая коэффициента масштабирования).

Строго говоря, процедура исключения весов тоже не является корректной формой регуляризации сложности многослойного персептрона, так как она также не вписывается в определение (4.95). Тем не менее при правильном подборе параметра w_0 она позволяет лучше корректировать отдельные веса сети, чем метод снижения весов [494].

Сглаживающая аппроксимация

В [752] для многослойного персептрона с одним скрытым слоем и одним нейроном в выходном слое предложено следующее слагаемое штрафа за сложность:

$$\mathbf{E}_c(\mathbf{w}) = \sum_{j=1}^{M} w_{oj}^2 \, \|\mathbf{w}_j\|^p, \tag{4.98}$$

где w_{oj} — веса выходного слоя; $\mathbf{w}_j$ — весовой вектор j-го нейрона скрытого слоя, а показатель степени p определяется выражением

$$p = \begin{cases} 2k - 1 \text{ для глобального сглаживания,} \\ 2k \text{ для локального сглаживания,} \end{cases} \tag{4.99}$$

где k — порядок дифференцирования функции $F(\mathbf{x}, \mathbf{w})$ по $\mathbf{x}$.

Процедура сглаживающей аппроксимации в задаче регуляризации сложности многослойного персептрона оказалась более строгой, чем снижение или исключение весов. В отличие от ранее описанных методов она выполняет следующие задачи.

1. Разделяет роли синаптических весов нейронов скрытого и выходного слоев.
2. Отслеживает взаимодействие этих двух множеств весов.

Однако при этом она имеет более сложную форму, чем снижение и исключение весов, следовательно, более требовательна к ресурсам из-за своей вычислительной сложности.

Упрощение структуры сети на основе Гессиана

Главной идеей этого подхода к упрощению структуры сети (усечению сети) является использование информации о вторых производных поверхности ошибок для обеспечения компромисса между сложностью сети и величиной ошибки обучения. В частности, для аналитического прогнозирования эффекта изменения синаптических весов строится локальная модель поверхности ошибок. Создание такой модели начинается с локальной аппроксимации функции стоимости $\mathbf{E}_{\text{av}}$ с помощью *ряда Тейлора* (Taylor series) в окрестности выбранной точки:

$$\mathbf{E}_{\text{av}}(\mathbf{w}+\Delta\mathbf{w}) = \mathbf{E}_{\text{av}}(\mathbf{w}) + \mathbf{g}^T(\mathbf{w})\Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^T\mathbf{H}\Delta\mathbf{w} + O(\|\Delta\mathbf{w}\|^3), \qquad (4.100)$$

где $\Delta\mathbf{w}$ — возмущение, применяемое к данной точке $\mathbf{w}$; $\mathbf{g}(\mathbf{w})$ — вектор градиента, вычисленный в точке $\mathbf{w}$. Гессиан тоже вычисляется в точке $\mathbf{w}$, и, таким образом, корректно применять обозначение $\mathbf{H}(\mathbf{w})$. Это не было сделано в формуле (4.100) для упрощения выкладок.

Требуется определить множество параметров, исключение которых из многослойного персептрона вызовет минимальное увеличение функции стоимости $\mathbf{E}_{\text{av}}$. Для того чтобы решить эту задачу на практике, необходимо выполнить следующее.

1. *Внешняя аппроксимация* (external approximation). Предполагается, что параметры удаляются из сети только после сходимости процесса обучения (т.е. после *полного обучения сети* (fully trained)). Следствие этого допущения состоит в том, что некоторые значения параметров соответствуют локальным или глобальным минимумам поверхности ошибки. В таком случае вектор градиента $\mathbf{g}$ можно считать равным нулю и слагаемое $\mathbf{g}^T\Delta\mathbf{w}$ в правой части (4.100) можно не учитывать. В противном случае мера выпуклости (которая будет определена позже) для данной задачи может быть определена неверно.
2. *Квадратичная аппроксимация* (quadratic approximation). Предполагается, что поверхность ошибок в окрестности глобального или локального минимума является приблизительно "квадратичной". Исходя из этого, в выражении (4.100) можно также не учитывать все слагаемые более высокого порядка.

Исходя из этих допущений, аппроксимация (4.100) упрощается до следующего вида:

$$\Delta\mathbf{E}_{\text{av}} = \mathbf{E}(\mathbf{w}+\Delta\mathbf{w}) - \mathbf{E}(\mathbf{w}) = \frac{1}{2}\Delta\mathbf{w}^T\mathbf{H}\Delta\mathbf{w}. \qquad (4.101)$$

Процедура *оптимальной степени повреждения* (optimal brain damage — OBD) [624] упрощает вычисления за счет допущения о том, что матрица Гессиана $\mathbf{H}$ является диагональной. Однако такое предположение не делается в процедуре *оптимальной хирургии мозга* (optimal brain surgeon — OBS) [428]. Таким образом, процедура OBS содержит процедуру OBD в качестве частного случая. Поэтому в дальнейшем мы будем следовать стратегии OBS.

Целью OBS является обнуление одного из синаптических весов для минимизации приращения функции стоимости $\mathbf{E}_{\text{av}}$ в (4.101). Пусть $w_i(n)$ — некоторый синаптический вес связи. Удаление этого веса эквивалентно выполнению условия

$$\Delta w_i + w_i = 0$$

или

$$\mathbf{1}_i^T\Delta\mathbf{w} + w_i = 0, \qquad (4.102)$$

где $\mathbf{1}_i$ — единичный вектор, все элементы которого равны нулю за исключением i-го, который равен единице. Теперь цель OBS можно определить в следующем виде [428].

Необходимо минимизировать квадратичную форму $\frac{1}{2}\Delta \boldsymbol{w}^T \mathbf{H} \Delta \boldsymbol{w}$ *по отношению к приращению вектора весов* $\Delta \boldsymbol{w}$*, принимая в качестве ограничения равенство нулю выражения* $\boldsymbol{I}_i^T \Delta \boldsymbol{w} + w_i$*. После этого требуется минимизировать результат относительно индекса* i*.*

Из этого определения вытекает наличие двух уровней минимизации. Одна процедура минимизации выполняется по векторам синаптических весов, оставшихся после обнуления i-го вектора весов. Вторая операция минимизации определяет, какой из векторов можно исключить.

Для решения этой *задачи условной оптимизации* (constrained optimization problem) в первую очередь нужно построить *Лагранжиан* (Lagrangian):

$$S = \frac{1}{2}\Delta \mathbf{w}^T \mathbf{H} \Delta \mathbf{w} - \lambda(\mathbf{1}_i^T \Delta \mathbf{w} + w_i), \tag{4.103}$$

где λ — *множитель Лагранжа* (Lagrange multiplyer). Дифференцируя Лагранжиан S по $\Delta \mathbf{w}$, применяя условие (4.102) и используя обратную матрицу, находим, что оптимальное приращение вектора весов $\mathbf{w}$ вычисляется по следующей формуле:

$$\Delta \mathbf{w} = -\frac{w_i}{[\mathbf{H}^{-1}]_{i,i}} \mathbf{H}^{-1} \mathbf{1}_i, \tag{4.104}$$

а соответствующее ему оптимальное значение Лагранжиана для элемента w_i — по формуле

$$S_i = \frac{w_i^2}{2\,[\mathbf{H}^{-1}]_{i,i}}, \tag{4.105}$$

где $\mathbf{H}^{-1}$ — матрица, обратная Гессиану $\mathbf{H}$; $[\mathbf{H}^{-1}]_{i,i}$ — элемент с индексом i, i этой обратной матрицы. Лагранжиан S_i, оптимизированный по $\Delta \mathbf{w}$, подлежащий ограничению при исключении i-го синаптического веса w_i, называется *степенью выпуклости* w_i. Выпуклость S_i описывает рост среднеквадратической ошибки (как меры эффективности), вызываемый удалением синаптического веса w_i. Обратите внимание, что выпуклость S_i пропорциональна квадрату синаптического веса w_i^2. Таким образом, вес с маленьким значением оказывает слабое влияние на среднеквадратическую ошибку. Однако из выражения (4.105) видно, что величина S_i также обратно пропорциональна диагональным элементам матрицы, обратной Гессиану. Таким образом, если величина $[\mathbf{H}^{-1}]_{i,i}$ мала, то даже небольшие веса могут оказывать существенное влияние на среднеквадратическую ошибку.

В процедуре OBS для удаления выбираются веса, соответствующие наименьшей степени выпуклости. Более того, соответствующие оптимальные значения модификации оставшихся весов вычисляются по формуле (4.104), т.е. веса изменяются вдоль направления i-го столбца матрицы, обратной Гессиану.

В [428] показано, что в некоторых тестовых задачах сети полученные в результате применения процедуры OBS оказались меньше, чем построенные с использованием процедуры снижения весов. В этой же работе показано, что в результате применения процедуры OBS к многослойному персептрону NETtalk, имеющему один скрытый слой и 18000 весов, сеть сократилась до 1560 весов. Это довольно впечатляющий результат. Сеть NETtalk, разработанная в [962], описывается в главе 13.

Вычисление матрицы, обратной Гессиану

Вычисление матрицы, обратной Гессиану, является основой процедуры OBS. Когда количество свободных параметров W сети становится большим, задача вычисления $\mathbf{H}^{-1}$ может стать неразрешимой. В этом разделе описывается управляемая процедура вычисления $\mathbf{H}^{-1}$, в предположении, что многослойный персептрон был полностью обучен, т.е. достиг некоторого локального минимума на поверхности ошибок [428].

Для упрощения выкладок предположим, что многослойный персептрон содержит единственный выходной нейрон. Тогда для данного множества обучения функцию стоимости можно выразить соотношением

$$\mathbf{E}_{\text{av}}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(d(n) - o(n))^2,$$

где $o(n)$ — реальная выходная реакция сети на подачу n-го примера; $d(n)$ — соответствующий желаемый отклик; N — общее количество примеров в множестве обучения. Сам выходной сигнал $o(n)$ можно представить в виде

$$o(n) = F(\mathbf{w}, \mathbf{x}),$$

где $F(\mathbf{w}, \mathbf{x})$ — функция отображения "вход-выход", реализуемая многослойным персептроном; $\mathbf{x}$ — входной вектор; $\mathbf{w}$ — вектор синаптических весов сети. Исходя из этого, первую производную функции стоимости по $\mathbf{w}$ можно выразить формулой

$$\frac{\partial \mathbf{E}_{\text{av}}}{\partial \mathbf{w}} = -\frac{1}{N}\sum_{n=1}^{N}\frac{\partial F(\mathbf{w}, \mathbf{x}(n))}{\partial \mathbf{w}}(d(n) - o(n)), \tag{4.106}$$

а вторую производную по $\mathbf{w}$, или матрицу Гессиана, представить в виде

$$\mathbf{H}(N) = \frac{\partial^2 \mathbf{E}_{\text{av}}}{\partial \mathbf{w}^2} = \frac{1}{N}\sum_{n=1}^{N}\left\{\left(\frac{\partial F(\mathbf{w}, \mathbf{x}(n))}{\partial \mathbf{w}}\right)\left(\frac{\partial F(\mathbf{w}, \mathbf{x}(n))}{\partial \mathbf{w}}\right)^T - \right.$$
$$\left. - \left(\frac{\partial^2 F(\mathbf{w}, \mathbf{x}(n))}{\partial \mathbf{w}^2}\right)(d(n) - o(n))\right\}. \tag{4.107}$$

В этой формуле явным образом выделена зависимость матрицы Гессиана от размера обучающего множества N.

Если сеть полностью обучена, т.е. функция стоимости достигла одного из своих локальных минимумов на поверхности ошибок, есть основания утверждать, что значение $o(n)$ достаточно близко к $d(n)$. При этом условии второе слагаемое можно не учитывать и аппроксимировать выражение (4.107) следующим:

$$\mathbf{H}(N) \simeq \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial F(\mathbf{w}, \mathbf{x}(n))}{\partial \mathbf{w}}\right)\left(\frac{\partial F(\mathbf{w}, \mathbf{x}(n))}{\partial \mathbf{w}}\right)^T. \tag{4.108}$$

Для упрощения записи введем вектор размерности $W \times 1$:

$$\boldsymbol{\xi}(n) = \frac{1}{\sqrt{N}}\frac{\partial F(\mathbf{w}, x(n))}{\partial \mathbf{w}}, \tag{4.109}$$

который можно вычислить с помощью процедуры, представленной в разделе 4.10. Исходя из этого, выражение (4.108) можно переписать в рекурсивной форме:

$$\mathbf{H}(n) = \sum_{k=1}^{n}\boldsymbol{\xi}(k)\boldsymbol{\xi}^T(k) = H(n-1) + \boldsymbol{\xi}(n)\boldsymbol{\xi}^T(n), \quad n = 1, 2, \ldots, N. \tag{4.110}$$

Эту рекурсивную запись удобно применить в так называемой *лемме об инвертировании матриц* (matrix inversion lemma), известной также под названием *равенства Вудбурри* (Woodburry equality).

Пусть $\mathbf{A}$ и $\mathbf{B}$ — две положительно определенные матрицы, удовлетворяющие соотношению

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}\mathbf{C}^T,$$

где $\mathbf{C}$ и $\mathbf{D}$ — две другие матрицы. Согласно лемме об инвертировании матриц, матрица, обратная $\mathbf{A}$, определяется соотношением

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{B}.$$

Для задачи, определяемой соотношением (4.110), имеем:

$$\begin{aligned}\mathbf{A} &= \mathbf{H}(n),\\ \mathbf{B}^{-1} &= \mathbf{H}(n-1),\\ \mathbf{C} &= \boldsymbol{\xi}(n),\\ \mathbf{D} &= 1.\end{aligned}$$

Применяя лемму об инвертировании матриц, можно вывести требуемую рекурсивную формулу вычисления матрицы, обратной Гессиану:

$$\mathbf{H}^{-1}(n) = \mathbf{H}^{-1}(n-1) - \frac{\mathbf{H}^{-1}(n-1)\boldsymbol{\xi}(n)\boldsymbol{\xi}^{T}(n)\mathbf{H}^{-1}(n-1)}{1+\boldsymbol{\xi}^{T}(n)\mathbf{H}^{-1}(n-1)\boldsymbol{\xi}(n)}. \tag{4.111}$$

Обратите внимание, что знаменатель в формуле (4.111) является скаляром, следовательно, вычисление обратной матрицы не составляет труда. Имея последнее значение $\mathbf{H}^{-1}(n-1)$, можно вычислить значение $\mathbf{H}^{-1}(n)$ на основе n-го примера, представленного вектором $\boldsymbol{\xi}(n)$. Рекурсивные вычисления продолжаются до тех пор, пока не будет обработано все множество из N примеров. Для инициализации этого алгоритма необходимо задать достаточно большое начальное приближение $\mathbf{H}^{-1}(0)$, так как на каждом шаге рекурсии элементы матрицы будут только убывать. Это требование можно удовлетворить с помощью следующего значения матрицы:

$$\mathbf{H}^{-1}(0) = \delta^{-1}\mathbf{I}, \tag{4.112}$$

где δ — достаточно малое положительное число; $\mathbf{I}$ — единичная матрица. Такая форма исходного значения гарантирует, что матрица $\mathbf{H}^{-1}(n)$ всегда будет положительно определенной. Влияние δ прогрессивно уменьшается по мере подачи все большего количества примеров.

Алгоритм оптимальной хирургии мозга (OBS) в сжатом виде представлен в табл. 4.6 [428].

4.16. Преимущества и ограничения обучения методом обратного распространения

Алгоритм обратного распространения является самым популярным среди алгоритмов обучения многослойного персептрона с учителем. По существу он представляет собой градиентный метод, а *не* метод оптимизации. Процедура обратного распространения обладает двумя основными свойствами.

ТАБЛИЦА 4.6. Алгоритм оптимальной хирургии мозга

1. Минимизируем среднеквадратическую ошибку в процессе обучения данного многослойного персептрона.
2. Используем процедуру, описанную в разделе 4.10, для вычисления вектора
$$\boldsymbol{\xi}(n) = \frac{1}{\sqrt{N}} \frac{\partial F(\mathbf{w}, \mathbf{x}(n))}{\partial \mathbf{w}},$$
где $F(\mathbf{x},\mathbf{w})$ — отображение "вход-выход", реализуемое многослойным персептроном с вектором синаптических весов $\mathbf{w}$ и вектором входного сигнала $\mathbf{x}$.
3. С помощью рекурсивного выражения (4.111) вычисляем матрицу, обратную Гессиану ($\mathbf{H}^{-1}$).
4. Находим некоторое i, соответствующее наименьшей степени выпуклости
$$S_i = \frac{w_i^2}{2[\mathbf{H}^{-1}]_{i,i}},$$
где $[\mathbf{H}^{-1}]_{i,i}$ — i,i-й элемент этой матрицы $\mathbf{H}^{-1}$. Если степень выпуклости S_i гораздо меньше, чем среднеквадратическая ошибка $\mathbf{E}_{\text{av}}$, то данный синаптический вес w_i удаляется, и мы снова переходим к шагу 4. В противном случае переходим к шагу 5.
5. Изменяем все синаптические веса сети, применяя приращение
$$\Delta \mathbf{w} = -\frac{w_i}{[\mathbf{H}^{-1}]_{i,i}} \mathbf{H}^{-1} \mathbf{1}_i,$$
переходим к шагу 2.
6. Процесс вычисления прекращается, если больше не существует весов, которые можно удалить из сети, существенно не увеличив среднеквадратическую ошибку (в этот момент сеть желательно обучить заново).

- Во-первых, ее *легко* просчитать локально.
- Во-вторых, она реализует *стохастический* градиентный спуск в пространстве весов (при котором синаптические веса обновляются для каждого примера).

Эти два свойства обучения методом обратного распространения в контексте многослойного персептрона и определяют его преимущества и ограничения.

Связность

Алгоритм обратного распространения является примером *парадигмы связности* (connectionist paradigm), согласно которой возможности нейронной сети по обработке информации реализуются за счет локальных вычислений. Эта форма вычислительных ограничений называется *ограничением локальности* (locality constraint) в том смысле, что вычисления в каждом нейроне, на который воздействуют другие нейроны, выполняются обособленно. Использование локальных вычислений в контексте искусственных нейронных сетей объясняется тремя принципиальными причинами.

1. Искусственные нейронные сети, осуществляющие локальные вычисления, часто выступают в качестве модели биологических нейронных сетей.
2. Использование локальных вычислений вызывает плавное снижение производительности при сбоях в аппаратном обеспечении. Это обеспечивает отказоустойчивость таких систем.
3. Локальные вычисления хорошо подходят для параллельной архитектуры, которая применяется в качестве эффективного метода реализации искусственных нейронных сетей.

Рассматривая эти три причины в обратном порядке, можно сказать, что третья вполне объяснима в случае обучения методом обратного распространения. В частности, этот алгоритм был успешно реализован многими исследователями в параллельных компьютерах, а архитектура VLSI была специально разработана для аппаратной реализации многослойного персептрона [411], [412]. Пункт 2 описывает некоторую меру предосторожности, принимаемую при реализации алгоритма обратного распространения [557]. Что же касается пункта 1, относящегося к биологическому правдоподобию алгоритма обратного распространения, то он может быть серьезно оспорен на следующем основании [232], [976], [1021].

1. Синаптические связи между нейронами многослойного персептрона могут быть как возбуждающими, так и тормозящими. Однако в реальной нервной системе каждый нейрон может играть только одну из этих ролей. Это одно из самых нереалистичных допущений, принимаемых при создании искусственных нейронных сетей.
2. В многослойном персептроне гормональные и прочие типы глобальных связей не учитываются. В реальной нервной системе эти типы глобальных связей играют важную роль для реализации таких функций, как внимание, обучение и раздражение.
3. При обучении методом обратного распространения синаптические веса изменяются только на основе предсинаптической активности и сигнала ошибки (обучения), независимо от постсинаптической активности. Очевидно, что нейробиология доказывает обратное.
4. В контексте нейробиологии реализация обучения методом обратного распространения требует быстрой обратной передачи информации по аксону. Очень маловероятно, чтобы такие действия на самом деле происходили в мозге.
5. Обучение методом обратного распространения предполагает наличие “учителя”, который в контексте работы мозга должен представляться отдельным множеством нейронов с неизвестными свойствами. Существование таких нейронов неправдоподобно с точки зрения биологии.

6. Тем не менее эта несогласованность с результатами нейробилогии не умаляют инженерной ценности обучения методом обратного распространения как средства обработки информации. Это было уже доказано множеством успешных применений в различных областях деятельности человека, в том числе и при моделировании нейробиологических явлений (например, в [893]).

Извлечение признаков

Как уже говорилось в разделе 4.9, скрытые нейроны многослойного персептрона, обучаемого методом обратного распространения, играют роль детекторов признаков. Это важное свойство многослойного персептрона было использовано совершенно неожиданным образом: было предложено использовать многослойный персептрон в качестве *репликатора* (replicator) или *карты идентичности* (identity map) [216], [915]. На рис. 4.23 показано, как этого можно достичь в случае многослойного персептрона с одним скрытым слоем. Архитектура сети удовлетворяет следующим структурным требованиям (см. рис. 4.23, *а*).

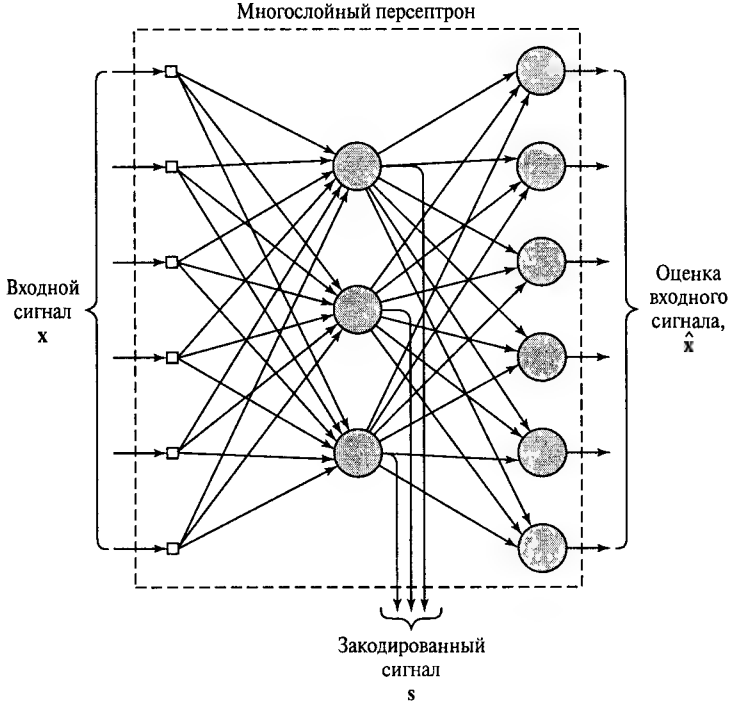
- Входной и выходной слои имеют одинаковый размер — m.
- Размер скрытого слоя M меньше значения m.
- Сеть является полносвязной.

Образ $\mathbf{x}$ синхронно подается во входной слой в качестве возбудителя и в выходной слой в качестве желаемого отклика. Фактический отклик выходного слоя $\hat{\mathbf{x}}$ расценивается как "оценка" вектора $\mathbf{x}$. Сеть обучается с помощью алгоритма обратного распространения, при этом в качестве сигнала ошибки выступает вектор ошибки оценивания $(\mathbf{x} - \hat{\mathbf{x}})$ (см. рис. 4.23, *б*). Можно сказать, что обучение производится без учителя. С помощью специальной структуры, встроенной в конструкцию многослойного персептрона, сеть осуществляет идентификацию посредством своего скрытого слоя. *Закодированная* (encoded) версия входного образа, обозначаемая символом $\mathbf{s}$, является выходным сигналом скрытого слоя (см. рис. 4.23, *а*). В результате полностью обученный многослойный персептрон выполняет роль системы кодирования. Для того чтобы воссоздать оценку $\hat{\mathbf{x}}$ исходного входного сигнала $\mathbf{x}$ (т.е. выполнить *декодирование*), закодированный сигнал передается скрытому слою сети репликации (см. рис. 4.23, *в*). В результате эта часть сети выполняет роль декодера. Чем меньше размер скрытого слоя M по сравнению с m, тем более эффективна эта система в качестве средства сжатия данных[12].

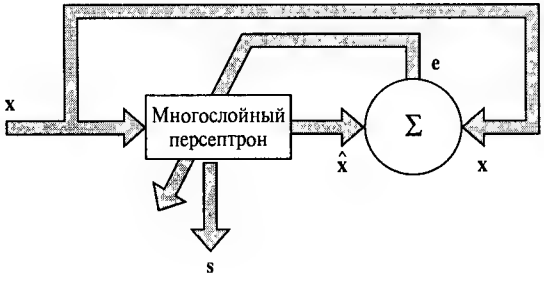
---

[12] В [446] описывается нейронная сеть репликации, представлеиная в форме многослойного персептрона с тремя скрытыми и одним выходным слоем.

Функции активации всех нейронов второго и четвертого (скрытых) слоев описываются функцией гиперболического тангенса $\varphi^{(2)}(v) = \varphi^{(4)}(v) = \tanh(v)$, где v — индуцированное локальное поле нейронов этих слоев.



а)



б)

Функции активации всех нейронов среднего (скрытого) слоя описываются следующей функцией:

$$\varphi^{(3)}(v) = \frac{1}{2} + \frac{1}{2(N-1)} \sum_{j=1}^{N-1} \tanh\left(a\left(v - \frac{j}{N}\right)\right),$$

где a — коэффициент усиления (gain parameter); v — индуцированное локальное поле нейронов этого слоя.

Функция $\varphi^{(3)}$ описывает *гладкую ступенчатую* (staircase) функцию активации с N ступеньками, выполняющую квантование вектора выходов соответствующих нейронов на $K = N^n$ элементов, где n – количество нейронов среднего скрытого слоя.

Нейроны выходного слоя являются линейными и используют функцию активации $\varphi^{(5)}(v) = v$. На основе этой структуры нейронной сети Хехт-Нильсен доказал теорему об оптимальном сжатии данных для произвольных входных векторов.

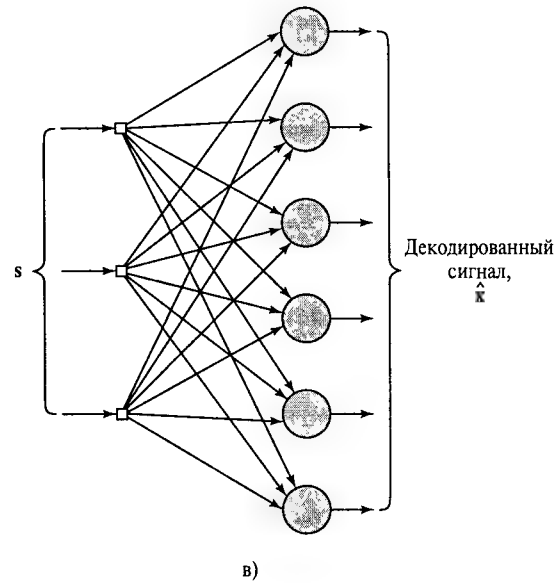


Рис. 4.23. Сеть репликации (карта идентичности) с одним скрытым слоем (а), используемым в качестве системы кодирования; блочная диаграмма обучения сети репликации с учителем (б) и часть сети репликации, используемая в качестве декодера (в)

Аппроксимация функций

Многослойный персептрон, обученный согласно алгоритму обратного распространения, используется в качестве *вложенной сигмоидальной схемы*, которую для случая одного выхода можно записать в следующем компактном виде:

$$F(\mathbf{x}, \mathbf{w}) = \varphi\left(\sum_k w_{ok}\varphi\left(\sum_j w_{kj}\varphi\left(\ldots\varphi\left(\sum_i w_{li}x_i\right)\right)\right)\right), \qquad (4.113)$$

где $\varphi(\cdot)$ — сигмоидальная функция активации общего вида; w_{ok} — синаптический вес связи между нейроном k последнего скрытого слоя и единственным выходным нейроном o, и т.д. для всех остальных синаптических весов; x_i — i-й элемент входного вектора $\mathbf{x}$. Вектор $\mathbf{w}$ содержит полное множество синаптических весов, упорядоченное сначала по слоям, затем по нейронам каждого отдельного слоя и, наконец, по синапсам отдельных нейронов. Представление в виде вложенной нелинейной функции, описываемой выражением (4.113), не является традиционным для классической теории аппроксимации. Как следует из раздела 4.13, это соотношение описывает *универсальный аппроксиматор* (universal approximator).

Использование метода обратного распространения открывает еще одно полезное свойство в контексте аппроксимации. Интуитивно понятно, что для многослойного персептрона с гладкой функцией активации производные результирующей функции должны аппроксимировать производные исходного реализуемого персептроном отображения "вход-выход". Доказательство этого результата представлено в [485].

В частности, в этой работе показано, что многослойный персептрон может аппроксимировать функции, не дифференцируемые в классическом смысле. При этом в случае кусочно-дифференцируемых функций он обеспечивает обобщенные производные. Полученные в этой статье результаты обеспечили ранее отсутствовавшее обоснование применения многослойного персептрона в приложениях, требующих аппроксимации функций и их производных.

Вычислительная эффективность

Вычислительная сложность (computational complexity) алгоритма обычно измеряется в терминах количества операций сложения, умножения и хранения, используемых в его реализации (см. главу 2). Алгоритм обучения считается *вычислительно эффективным* (computationally efficient), если его вычислительная сложность является *полиномиальной* (polynomial) по отношению к числу настраиваемых параметров, которые должны изменяться на каждой итерации. Основываясь на этом, можно утверждать, что алгоритм обратного распространения является вычислительно эффективным. В частности, при использовании этого алгоритма для обучения многослойного персептрона, содержащего W синаптических весов (включая пороги), его вычислительная сложность линейно зависит от W. Это важное свойство алгоритма обратного распространения может быть проверено с помощью прямого подсчета количества операций, осуществляемых при прямом и обратном проходах, описанных в разделе 4.5. При прямом проходе синаптические веса задействованы только для вычисления индуцированных локальных полей отдельных нейронов сети. Из формулы (4.44) видно, что эти вычисления линейны по синаптическим весам. При обратном проходе синаптические веса используются при вычислении локальных градиентов скрытых нейронов и при изменении самих синаптических весов, что следует из формул (4.46) и (4.47) соответственно. Таким образом, все вычисления линейны по синаптическим весам сети. Отсюда следует вывод, что вычислительная сложность алгоритма обратного распространения является линейной по W, т.е. составляет $O(W)$.

Анализ чувствительности

Еще одним преимуществом обучения методом обратного распространения с вычислительной точки зрения является эффективность анализа чувствительности отображения "вход-выход" с помощью этого алгоритма. Под *чувствительностью* (sensitivity) функции F по отношению к некоторому ее параметру ω понимается величина

$$S_{\omega}^{F} = \frac{\partial F / F}{\partial \omega / \omega}. \tag{4.114}$$

Рассмотрим многослойный персептрон, обучаемый с помощью алгоритма обратного распространения. Пусть функция $F(\mathbf{w})$ описывает осуществляемое сетью отображение входа на выход, а $\mathbf{w}$ — это вектор всех синаптических весов сети (включая пороги). В разделе 4.10 было показано, что частные производные функции $F(\mathbf{w})$ по всем элементам вектора весов $\mathbf{w}$ могут быть вычислены достаточно эффективно. В частности, исследуя выражения (4.81)–(4.83) совместно с формулой (4.114), можно увидеть, что сложность вычисления каждой отдельной частной производной линейно зависит от W — общего количества весов, содержащихся в сети. Эта линейность сохраняется всюду, где рассматриваемые синаптические веса присутствуют в цепочке вычислений.

Робастность

В главе 3 уже указывалось, что алгоритм LMS является робастным, т.е. возмущения с малой энергией незначительно влияют на ошибку оценивания. Если рассматриваемая модель наблюдений является линейной, алгоритм LMS представляет собой H^{∞}-оптимальный фильтр [426], [427]. Это значит, что алгоритм LMS минимизирует возрастание *максимальной энергии* (maximum energy gain) ошибки оценивания в результате возмущений.

С другой стороны, в [429] показано, что в случае нелинейной модели алгоритм обратного распространения является *локальным* H^{∞}-оптимальным фильтром. Под термином "локальный" здесь подразумевается то, что исходные значения вектора весов, используемые в алгоритме обратного распространения, должны быть достаточно близки к оптимальному значению вектора весов $\mathbf{w}^*$, чтобы гарантировать непопадание алгоритма в неверный локальный минимум. В концептуальном смысле приятно осознавать, что алгоритмы LMS и обратного распространения относятся к одному и тому же классу H^{∞}-оптимальных фильтров.

Сходимость

В алгоритме обратного распространения используется *текущая оценка* (instantaneous estimate) градиента поверхности ошибок в пространстве весов. Этот алгоритм является *вероятностным* по своей природе. Это значит, что движение в направлении минимума на поверхности ошибок происходит по зигзагообразной траектории. В самом деле, алгоритм обратного распространения является одной из реализаций статистического метода, известного под названием *стохастической аппроксимации* (stochastic approximation) и предложенного в [892]. Данный метод отличается медленной сходимостью, что может объясняться двумя причинами [505].

1. Поверхность ошибок является достаточно плоской вдоль направления некоторого весового коэффициента, поэтому частная производная в этом направлении невелика по амплитуде. В этом случае коррекции, применяемые к синаптическим весам, также являются малыми, и, следовательно, для достижения алгоритмом минимальной ошибки может потребоваться достаточно большое количество итераций. Если же поверхность ошибок по некоторым измерениям достаточно искривлена, то в этих направлениях частные производные велики по амплитуде. Значит, синаптические веса на каждом шаге будут подвергаться значительной коррекции. Это может привести к прохождению мимо точки локального минимума на поверхности ошибок.
2. Направление антиградиента (т.е. производная функции стоимости по вектору весов, взятая с противоположным знаком) может не соответствовать направлению к точке минимума на поверхности ошибок. Тогда коррекция весовых коэффициентов может привести к движению алгоритма в неправильном направлении.

Следовательно, скорость сходимости метода обратного распространения относительно невелика, что, в свою очередь, ведет к большой продолжительности вычислений. Согласно экспериментальным исследованиям, описанным в [919], скорость локальной сходимости алгоритма обратного распространения является *линейной*. Авторы работы объяснили это тем, что матрица Якобиана, а следовательно и матрица Гессиана, имеют неполный ранг. Эти наблюдения являются следствием *плохой обусловленности* самой задачи обучения нейронной сети. В [919] линейную локальную сходимость алгоритма обратного распространения авторы прокомментировали следующим образом.

- Преимуществом алгоритма обратного распространения по сравнению с другими методами более высокого порядка является то, что последние, при незначительном ускорении процесса сходимости, требуют существенно больше вычислительных ресурсов.
- Крупномасштабные задачи обучения нейронных сетей являются настолько трудными для реализации, что для них не существует приемлемой стратегии обучения с учителем и зачастую требуется предварительная обработка данных.

Проблема сходимости более подробно рассматривается в разделе 4.17, а вопрос предварительной обработки данных — в главе 8.

Локальные минимумы

Характерной особенностью поверхности ошибок, которая влияет на производительность алгоритма обратного распространения, является наличие не только глобального, но и *локальных минимумов* (local minima) (т.е. изолированных впадин). Так как в процессе обучения методом обратного распространения направление спуска опреде-

ляется направлением вектора градиента (наибольшего наклона поверхности), всегда существует риск попадания в точку локального минимума, в которой любое приращение синаптических весов будет приводить к увеличению функции стоимости. Однако при этом в другой области пространства весовых коэффициентов могут существовать другие множества синаптических весов, для которых функция стоимости имеет меньшее значение, чем в данной точке локального минимума. Понятно, что остановка процесса обучения в точке локального минимума является крайне нежелательной, если эта точка находится выше точки глобального минимума.

Проблема локальных минимумов при обучении методом обратного распространения описывается в заключение расширенного издания классической книги Минского и Пейперта [744], где основное внимание фокусировалось на обсуждении двухтомника [912], посвященного *параллельным распределенным вычислениям* (parallel distributed processing). В главе 8 упомянутой книги утверждается, что попадание в точку локального минимума редко встречается в практических приложениях алгоритма обратного распространения. Минский и Пейперт отвергли это утверждение, указав, что вся история распознавания образов доказывает обратное. В [370] описывается простой пример, в котором, несмотря на потенциальную обучаемость сети (с одним скрытым слоем) решению задачи классификации нелинейно-разделяемым множеств, обучение методом обратного распространения прекращалось в точке локального минимума[13].

Масштабирование

В принципе, нейронные сети, подобные многослойному персептрону, обучаемому с помощью алгоритма обратного распространения, создают потенциал для создания универсальных вычислительных машин. Однако, для того чтобы такой потенциал был реализован в полной мере, нужно преодолеть проблему *масштабирования* (scaling). Суть этой проблемы состоит в сохранении эффективности сети по мере увеличения и сложности поставленной задачи. Мерой в данном вопросе выступает время обучения, необходимое для достижения сетью наилучших показателей в обобщении.

[13] Изначально существовала потребность в создании теоретической базы для обучения методом обратного распространения, учитывающей и объясняющей проблему локальных минимумов. Эта задача была сложной для выполнения. Тем не менее в литературе по данному вопросу наблюдался некоторый прогресс. В 1989 году в [87] была рассмотрена задача обучения для многослойных сетей прямого распространения с линейными функциями активации на основе метода обратного распространения. Главным результатом этой работы было утверждение, что поверхность ошибок имеет единственный минимум, соответствующий ортогональной проекции на подпространство, порожденное первыми наибольшими собственными числами матрицы ковариации, связанной с обучающими примерами. Все остальные критические точки поверхности являются седловыми. В 1992 году в [370] был рассмотрен более общий случай обучения методом обратного распространения для сети, содержащей нелинейные нейроны. Основным результатом этой работы является утверждение, что для линейно-разделимых образов гарантируется сходимость к оптимальному решению (т.е. к глобальному минимуму). Это достигается с помощью кумулятивного (пакетного) режима обучения методом обратного распространения. При этом сеть в части обобщения новых примеров превосходила возможности персептрона Розенблатта.

Среди возможных способов измерения размера или сложности вычислительных задач наиболее полезной и важной мерой, по мнению автора, является порядок предиката, определенный в [744], [745].

Для того чтобы объяснить, что подразумевается под предикатом, опишем функцию $\psi(X)$, которая может принимать только два значения. Обычно под этими двумя значениями подразумеваются числа 0 и 1. Однако, приравнивая эти две величины к логическим значениям TRUE и FALSE, функцию $\psi(X)$ можно рассматривать как *предикат* (predicate), т.е. переменную величину, значение истинности которой зависит от выбора аргумента X. Например, функцию $\psi(X)$ можно определить в виде

$$\psi_{\text{круга}}(X) = \begin{cases} 1, & \text{если фигура } X - \text{круг,} \\ 0, & \text{если фигура } X - \text{не круг.} \end{cases} \tag{4.115}$$

На основе идеи предиката в [1050] предложен эмпирический алгоритм обучения многослойного персептрона вычислению функции четности на основе метода обратного распространения. *Функция четности* (parity function) является булевым предикатом, определяемым выражением

$$\psi_{\text{четности}}(X) = \begin{cases} 1, & \text{если } |X| - \text{четное число,} \\ 0 & - \text{в остальных случаях,} \end{cases} \tag{4.116}$$

порядок которого равен количеству входов. Эксперименты показали, что время, необходимое сети для обучения вычислению функции четности, растет экспоненциально в зависимости от количества входов (т.е. порядка предиката) и что планы использования алгоритма обратного распространения для обучения произвольным сложным функциям чересчур оптимистичны.

Общеизвестно, что для многослойного персептрона нежелательно, чтобы сеть была полносвязной. В этом контексте можно поставить следующий вопрос: как лучше разместить синаптические связи сети неполносвязного многослойного персептрона? Этот вопрос не играет особой роли в небольших приложениях, но является жизненно важным при решении масштабных реальных задач на основе обучения методом обратного распространения.

Одним из эффективных методов ослабления проблемы масштабирования является взгляд в корень задачи (иногда с помощью аналогий из области нейробиологии) и внесение в архитектуру многослойного персептрона правдоподобных конструктивных решений. В частности, архитектура сети и ограничения, реализуемые синаптическими весами, должны задаваться с учетом априорной информации об объекте моделирования. Эта стратегия описывается в разделе 4.19 на примере решения задачи оптического распознавания символов.

4.17. Ускорение сходимости процесса обучения методом обратного распространения

В предыдущих разделах описывались основные причины, вызывающие низкую скорость сходимости алгоритма обратного распространения. В этом разделе будут предложены некоторые эвристические наработки, которые позволяют ускорить сходимость алгоритма с помощью настройки параметра скорости обучения. Приведем эти эвристики [505].

Эвристика 1. Любой настраиваемый параметр сети, входящий в функцию стоимости, должен иметь свое значение коэффициента скорости обучения.

Отсюда можно заключить, что алгоритм обратного распространения может иметь низкую скорость сходимости из-за того, что фиксированный параметр скорости обучения подходит не для всех областей поверхности ошибок. Другими словами, если параметр интенсивности обучения подходит для корректировки одного синаптического веса, то это совсем не означает, что он подойдет для настройки другого. Эвристика 1 учитывает этот факт и предлагает назначать каждому настраиваемому синаптическому весу (параметру) сети отдельное значение коэффициента скорости обучения.

Эвристика 2. Любой параметр скорости обучения должен варьироваться для различных итераций.

Поверхность ошибок в разных областях имеет разную динамику даже в направлении одного весового коэффициента. Для того чтобы учесть эти изменения, согласно эвристике 2, параметры обучения должны варьироваться на разных итерациях. Интересно, что эта эвристика хорошо обоснована для случая линейных процессорных элементов [684].

Эвристика 3. Если производная функции стоимости по отдельному синаптическому весу на нескольких последовательных итерациях имеет один и тот же знак, то значение параметра скорости обучения для данного веса должно увеличиваться.

Текущая точка в пространстве весов может лежать на относительно пологом участке вдоль некоторого направления весовых коэффициентов. Это, в свою очередь, может отразиться на производной функции стоимости (т.е. на градиенте поверхности ошибки) по данному направлению, которая в течение нескольких последовательных итераций будет иметь один и тот же алгебраический знак и, таким образом, указывать на одно и то же направление. Эвристика 3 утверждает, что в такой ситуации количество итераций, необходимое для прохождения пологой части поверхности ошибок, может быть уменьшено за счет соответствующего увеличения значения параметра скорости обучения.

Эвристика 4. Если производная функции стоимости по отдельному синаптическому весу на нескольких последовательных итерациях имеет разные знаки, то значение параметра скорости обучения для данного веса должно уменьшаться.

Текущая точка в пространстве весов может лежать на участке поверхности ошибок, содержащем множество выпуклостей и впадин (т.е. на достаточно искривленном участке), что может привести к изменению знака производной по некоторому весу на последовательных итерациях. Согласно эвристике 4, во избежание осцилляции параметр скорости обучения для данного веса следует уменьшить.

Естественно, введение зависимости параметра обучения от времени и конкретного синаптического веса, следующее из этих эвристик, приводит к фундаментальному изменению алгоритма обратного распространения. В частности, модифицированный таким образом алгоритм уже не будет осуществлять поиск методом наискорейшего спуска. Корректировка конкретных синаптических весов будет основываться на частных производных поверхности ошибок по конкретным весам и на оценке кривизны поверхности ошибок в текущей точке относительно конкретных измерений пространства весов.

Более того, все четыре эвристики удовлетворяют ограничению локальности, являющемуся встроенной характеристикой алгоритма обратного распространения. К сожалению, наличие ограничения локальности сужает область применения этих эвристик, так как могут существовать такие поверхности, где они просто не будут работать. Тем не менее модификации алгоритма обратного распространения, разработанные в соответствии с этими эвристиками, имеют большое практическое значение[14].

4.18. Обучение с учителем как задача оптимизации

В этом разделе обучение с учителем рассматривается с позиций, в корне отличающихся от точки зрения, изложенной во всех предыдущих разделах. В частности, обучение с учителем многослойного персептрона будет рассмотрено как задача *численной оптимизации* (numerical optimizing). В этом контексте в первую очередь следует отметить, что поверхность ошибок многослойного персептрона является в высшей степени нелинейной функцией, зависящей от вектора синаптических весов $\mathbf{w}$. Пусть $\mathbf{E}_{av}(\mathbf{w})$ — функция стоимости, усредненная на множестве примеров обучения. Используя ряд Тейлора, функцию $\mathbf{E}_{av}(\mathbf{w})$ можно экстраполировать в окрестности текущей точки на поверхности ошибок $\mathbf{w}(n)$, например в соответствии с выражением (4.100).

---

[14] Модификация алгоритма обратного распространения, основанная на эвристиках 1–4, получила название правила обучения *delta-bar-delta* [505]. Реализация этого правила обучения может быть упрощена с помощью идеи, аналогичной методу повторного использоваиия градиеита (gradient reuse method) [439], [496].

В [925] описана процедура динамической самоадаптации для ускорения процесса обучения методом обратного распространения. В этой процедуре реализована следующая идея: значение коэффициента скорости обучения для предыдущего шага немного уменьшается и увеличивается; для обоих новых значений оценивается функция стоимости, после чего выбирается то значение коэффициента, для которого функция стоимости имеет меньшую величину.

Перепишем это выражение, вводя явную зависимость от n:

$$\begin{aligned}\mathbf{E}_{\text{av}}(\mathbf{w}(n)+\Delta\mathbf{w}(n)) =& \mathbf{E}_{\text{av}}(\mathbf{w}(n)) + \mathbf{g}^T(n)\Delta\mathbf{w}(n) + \\ &+ \frac{1}{2}\Delta\mathbf{w}^T(n)\mathbf{H}(n)\Delta\mathbf{w}(n) + O(\|\Delta\mathbf{w}\|^3),\end{aligned} \tag{4.117}$$

где $\mathbf{g}(n)$ — вектор локального градиента:

$$\mathbf{g}(n) = \left.\frac{\partial \mathbf{E}_{\text{av}}(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}(n)}, \tag{4.118}$$

$\mathbf{H}(n)$ — матрица Гессиана в данной точке:

$$\mathbf{H}(n) = \left.\frac{\partial^2 \mathbf{E}_{\text{av}}(\mathbf{w})}{\partial \mathbf{w}^2}\right|_{\mathbf{w}=\mathbf{w}(n)}. \tag{4.119}$$

Использование функции стоимости $\mathbf{E}_{av}(\mathbf{w})$, усредненной по множеству, подразумевает пакетный режим обучения.

В методе наискорейшего спуска, представленного в данном случае алгоритмом обратного распространения, корректировка $\Delta\mathbf{w}(n)$, применяемая к синаптическому весу $\mathbf{w}(n)$, определяется выражением

$$\Delta\mathbf{w}(n) = -\eta\mathbf{g}(n), \tag{4.120}$$

где η — параметр скорости обучения. В результате метод наискорейшего спуска работает на основе *линейной аппроксимации* функции стоимости в окрестности текущей точки $\mathbf{w}(n)$, при которой единственным источником информации о поверхности ошибок является градиент $\mathbf{g}(n)$. Такое ограничение обеспечивает значительное преимущество: простоту реализации. К сожалению, оно приносит и нежелательный результат — низкую скорость сходимости, которая может оказаться камнем преткновения, особенно в задачах большого объема. Включение слагаемого момента в уравнение корректировки вектора синаптических весов было грубой попыткой использования информации второго порядка о поверхности ошибок, что привнесло некоторое преимущество. Тем не менее его использование сделало процесс обучения более чувствительным.

Для того чтобы существенно улучшить скорость сходимости многослойного персептрона (по сравнению с алгоритмом обратного распространения), в процессе обучения приходится использовать информацию *более высокого порядка*. Этого можно добиться, привлекая *квадратичную аппроксимацию* (quadratic approximation) поверхности ошибок в окрестности текущей точки $\mathbf{w}(n)$. При этом оказывается (см. (4.117)), что оптимальное значение приращения $\Delta\mathbf{w}(n)$, применяемого к вектору синаптиче-

ских весов $\mathbf{w}(n)$, определяется следующей формулой:

$$\Delta\mathbf{w}^*(n) = \mathbf{H}^{-1}(n)\mathbf{g}(n), \tag{4.121}$$

где $\mathbf{H}^{-1}(n)$ — матрица, обратная Гессиану (в предположении, что таковая существует). Выражение (4.121) описывает сущность *метода Ньютона* (Newton's method). Если функция стоимости $\mathbf{E}_{\text{av}}(\mathbf{w})$ является квадратичной (т.е. третье и следующие слагаемые ряда Тейлора (4.117) равны нулю), то метод Ньютона сходится к оптимальному значению всего за одну итерацию. Однако на практике применение метода Ньютона для обучения с учителем многослойного персептрона усложняется следующими факторами.

- Требуется вычисление матрицы, обратной Гессиану ($\mathbf{H}^{-1}(n)$), что само по себе весьма неэффективно с точки зрения объема вычислений.
- Для того чтобы иметь возможность вычислить матрицу $\mathbf{H}^{-1}(n)$, матрица $\mathbf{H}(n)$ должна быть несингулярной. Если матрица Гессиана положительно определена, поверхность ошибок в окрестности точки $\mathbf{w}(n)$ является выпуклой. К сожалению, нет гарантии, что матрица Гессиана для поверхности ошибок многослойного персептрона всегда будет соответствовать этому требованию. Более того, существует потенциальная проблема, что матрица Гессиана будет иметь недостаточный ранг (т.е. не все столбцы матрицы $\mathbf{H}$ будут линейно-независимыми), что приведет к плохой обусловленности задачи обучения нейронной сети [919]. Это существенно усложняет вычислительную задачу.
- Если функция стоимости $\mathbf{E}_{\text{av}}(\mathbf{w})$ не является квадратичной, то нельзя гарантировать сходимость метода Ньютона, что делает его непригодным для обучения многослойного персептрона.

Для преодоления подобных проблем можно использовать *квазиньютоновские методы*, которые используют только оценки вектора градиента $\mathbf{g}$. В таких модификациях метода Ньютона положительно определенная оценка матрицы $\mathbf{H}^{-1}(n)$ вычисляется напрямую, без обращения самой матрицы. Используя такую оценку, квазиньютоновские методы обеспечивают движение вниз по склону поверхности ошибок. Однако вычислительная сложность при этом измеряется порядком $O(W^2)$, где W — размерность вектора $\mathbf{w}$. С вычислительной точки зрения квазиньютоновские методы являются непрактичными, за исключением случаев обучения нейросетей малой размерности. Описание квазиньютоновских методов будет представлено ниже в настоящем разделе.

Еще одним классом методов второго порядка является метод *сопряженных градиентов* (conjugate-gradient). Его можно рассматривать как нечто среднее между методом наискорейшего спуска и методом Ньютона. Использование метода сопряженных градиентов мотивировано желанием повысить обычно низкую скорость сходимости метода наискорейшего спуска при одновременном избежании вычислительных слож-

ностей, связанных с оценкой, хранением и инвертированием матрицы Гессиана в методе Ньютона. Среди всех методов оптимизации второго порядка метод сопряженных градиентов считается единственным пригодным для решения задач большого объема, т.е. тех, которые имеют сотни и тысячи настраиваемых параметров [298]. Таким образом, он хорошо подходит для обучения многослойного персептрона при решении задач аппроксимации функций, управлении и задач регрессии.

Метод сопряженных градиентов

Метод сопряженных градиентов (conjugate-gradient method) относится к классу методов оптимизации, известных под названием *методов сопряженных направлений* (conjugate-direction methods). Обсуждение этих методов начнем с рассмотрения задачи минимизации *квадратичной функции* (quadratic function):

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + c, \tag{4.122}$$

где $\mathbf{x}$ — вектор параметров размерности $W \times 1$; $\mathbf{A}$ — симметричная, положительно определенная матрица размерности $W \times W$; $\mathbf{b}$ — вектор размерности $W \times 1$; c — скаляр. Минимум квадратичной функции $f(\mathbf{x})$ достигается при единственном значении $\mathbf{x}$:

$$\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}. \tag{4.123}$$

Таким образом, минимизация функции $f(\mathbf{x})$ и решение системы линейных уравнений $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ являются эквивалентными задачами.

Для данной матрицы $\mathbf{A}$ множество ненулевых векторов $\mathbf{s}(0), \mathbf{s}(1), \ldots, \mathbf{s}(W-1)$ называется $\mathbf{A}$-*сопряженным* ($\mathbf{A}$-conjugate), если для всех $n \neq j$ выполняется следующее соотношение:

$$\mathbf{s}(n)\mathbf{A}\mathbf{s}(j) = 0 \text{ для всех } n \text{ и } j \text{ при условии } n \neq j. \tag{4.124}$$

Если матрица $\mathbf{A}$ является единичной, сопряженность является эквивалентом ортогональности.

Пример 4.1

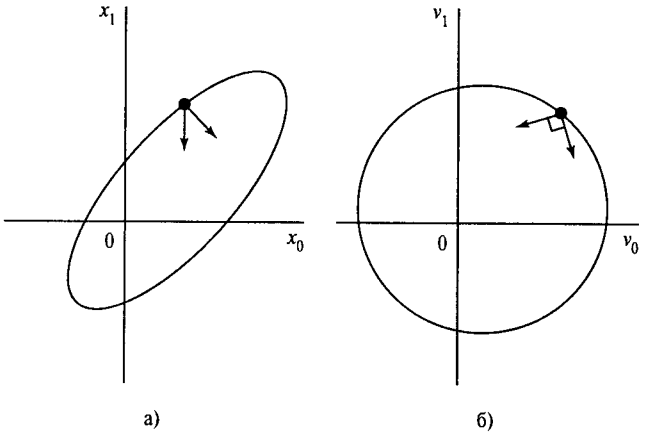
Для интерпретации А-сопряженных векторов рассмотрим двумерную задачу, проиллюстрированную на рис. 4.24, *а*. Показанная на этом рисунке эллиптическая траектория является графиком квадратичной функции $f(\mathbf{x})$ (см. (4.122)) от векторной переменной:

$$\mathbf{x} = [x_0, x_1]^T.$$

На рис. 4.24, *а* показана также пара векторов, определяющих направления, сопряженные по отношению к матрице $\mathbf{A}$. Предположим, определен новый вектор параметров $\mathbf{v}$, полученный из $\mathbf{x}$ с помощью преобразования

$$\mathbf{v} = \mathbf{A}^{1/2}\mathbf{x},$$

где $\mathbf{A}^{1/2}$ является квадратным корнем матрицы $\mathbf{A}$. Тогда эллиптическая траектория (см. рис. 4.24, *а*)



а) б)

Рис. 4.24. Интерпретация A-сопряженных векторов; эллиптическая траектория в двумерном пространстве весов (а) и преобразование эллиптической траектории в круговую (б)

преобразуется в круговую (рис. 4.24, *б*). При этом пара **A**-сопряженных векторов направлений (см. рис. 4.24) преобразуется в пару ортогональных векторов направлений (см. рис. 4.24, *б*). ■

Важным свойством **A**-сопряженных векторов является то, что они *линейно-независимы* (linearly independent). Это свойство можно доказать от противного. Пусть один из этих векторов, например $\mathbf{s}(0)$, можно представить как линейную комбинацию остальных $W-1$ векторов множества, т.е.

$$\mathbf{s}(0) = \sum_{j=1}^{W-1} \alpha_j \mathbf{s}(j).$$

Умножив матрицу **A** на вектор $\mathbf{s}(0)$ и затем скалярно умножив результат на вектор $\mathbf{s}^T(0)$, получим:

$$\mathbf{s}^T(0)\mathbf{A}\mathbf{s}(0) = \sum_{j=1}^{W-1} \alpha_j \mathbf{s}^T(0)\mathbf{A}\mathbf{s}(j) = 0.$$

Однако квадратичная форма $\mathbf{s}^T(0)\mathbf{A}\mathbf{s}(0)$ не может равняться нулю по двум причинам: во-первых, по определению матрица **A** является положительно определенной, а вектор $\mathbf{s}(0)$ — ненулевым. Отсюда следует, что **A**-сопряженные векторы $\mathbf{s}(0)$, $\mathbf{s}(1),\ldots,\mathbf{s}(W-1)$ не могут быть линейно-зависимыми, т.е. они являются линейно-независимыми.

Для данного множества **A**-сопряженных векторов $\mathbf{s}(0)$, $\mathbf{s}(1),\ldots,\mathbf{s}(W-1)$ существует соответствующий *метод сопряженных направлений* (conjugate direction method) минимизации функции квадратичной ошибки $f(\mathbf{x})$, определяемый выражением [125], [298], [682]

$$\mathbf{x}(n+1) = \mathbf{x}(n) + \eta(n)\mathbf{s}(n), n = 0, 1, \ldots, W-1, \qquad (4.125)$$

где $\mathbf{x}(0)$ — произвольный исходный вектор; $\eta(n)$ — скаляр, определяемый из соотношения

$$f(\mathbf{x}(n)+\eta(n)\mathbf{s}(n))=\min_{\eta} f(\mathbf{x}(n)+\eta\mathbf{s}(n)). \tag{4.126}$$

Процедура выбора параметра η сводится к минимизации функции $f(\mathbf{x}(n)+\eta\mathbf{s}(n))$ для некоторого фиксированного n. Это линейный поиск, представляющий одномерную задачу минимизации.

В свете выражений (4.124)–(4.126) можно сделать следующие наблюдения.

1. Поскольку $\mathbf{A}$-сопряженные векторы $\mathbf{s}(0)$, $\mathbf{s}(1),\ldots,\mathbf{s}(W-1)$ линейно-независимы, они формируют базис пространства векторов $\mathbf{w}$.
2. Формулы изменения (4.125) и линейной минимизации (4.126) приводят к хорошо знакомой формуле вычисления значения параметра скорости обучения:

$$\eta(n)=\frac{\mathbf{s}^T(n)\mathbf{A}\varepsilon(n)}{\mathbf{s}^T(n)\mathbf{A}\mathbf{s}(n)},\quad n=0,\ 1,\ldots,\ W-1, \tag{4.127}$$

где $\boldsymbol{\varepsilon}(n)$ — *вектор ошибки*, определяемый выражением

$$\boldsymbol{\varepsilon}(n)=\mathbf{x}(n)-\mathbf{x}^*. \tag{4.128}$$

3. Начиная с произвольной точки $\mathbf{x}(0)$, метод сопряженных направлений гарантирует нахождение оптимального решения $\mathbf{x}^*$ квадратного уравнения $f(\mathbf{x})=0$ не более чем за W итераций.

Принципиальное свойство метода сопряженных направлений можно описать следующим образом [125], [298], [682].

При последовательных итерациях метод сопряженных направлений минимизирует квадратичную функцию $f(\boldsymbol{x})$ в постепенно расширяющемся линейном пространстве векторов, содержащем точку глобального минимума этой функции.

В частности, для каждой итерации n оценка $\mathbf{x}(n+1)$ минимизирует функцию $f(\mathbf{x})$ в линейном пространстве векторов $\mathbf{D}_n$, проходящем через произвольную точку $\mathbf{x}(0)$ и задаваемом $\mathbf{A}$-сопряженными векторами $\mathbf{s}(0)$, $\mathbf{s}(1),\ldots,\mathbf{s}(W-1)$:

$$\mathbf{x}(n+1)=\arg\min_{\mathbf{x}\in\mathbf{D}_n} f(\mathbf{x}), \tag{4.129}$$

где пространство $\mathbf{D}_n$ определяется формулой

$$\mathbf{D}_n = \left\{ \mathbf{x}(n) | \mathbf{x}(n) = \mathbf{x}(0) + \sum_{j=0}^{n} \eta(j)\mathbf{s}(j) \right\}. \tag{4.130}$$

Для обеспечения работоспособности метода сопряженных направлений требуется определить множество **A**-сопряженных векторов $\mathbf{s}(0)$, $\mathbf{s}(1),\ldots,$ $\mathbf{s}(W-1)$. В частном случае этот метод называют *методом сопряженных градиентов* (conjugate-gradient method)[15]. Последовательные векторы направлений генерируются как **A**-сопряженные версии последовательных векторов градиента квадратичной функции $f(\mathbf{x})$. Отсюда этот метод и получил свое название. Таким образом, за исключением итерации $n = 0$, множество векторов направлений $\{\mathbf{s}(n)\}$ заранее не известно. Оно формируется последовательно на каждой итерации метода.

Выберем в качестве направления наискорейшего спуска *резидуальную ошибку* (residual error)

$$\mathbf{r}(n) = \mathbf{b} - \mathbf{A}\mathbf{x}(n). \tag{4.131}$$

Далее будем использовать комбинацию векторов $\mathbf{r}(n)$ и $\mathbf{s}(n-1)$ следующим образом:

$$\mathbf{s}(n) = \mathbf{r}(n) + \beta(n)\mathbf{s}(n-1), n = 1, 2, \ldots, W-1, \tag{4.132}$$

где $\beta(n)$ — множитель масштабирования, который необходимо определить. Умножая обе части равенства на **A**, а затем скалярно умножая результат на вектор $\mathbf{s}(n-1)$, учитывая свойство **A**-сопряженности векторов направлений и решая полученное уравнение относительно $\beta(n)$, получим:

$$\beta(n) = -\frac{\mathbf{s}(n-1)\mathbf{A}\mathbf{r}(n)}{\mathbf{s}^T(n-1)\mathbf{A}\mathbf{s}(n-1)}. \tag{4.133}$$

Принимая во внимание выражения (4.132) и (4.133), несложно увидеть, что сформированные таким образом векторы $\mathbf{s}(0)$, $\mathbf{s}(1),\ldots,$ $\mathbf{s}(W-1)$ действительно являются **A**-сопряженными.

Процесс генерирования векторов направлений в соответствии с рекурсивным правилом (4.132) зависит от коэффициента $\beta(n)$. Формула (4.133), определяющая вектор $\beta(n)$, подразумевает точное знание матрицы **A**. С вычислительной точки зрения хоте-

---

[15] Классической работой, посвященной методу сопряженных градиентов, считается [454]. Обсуждение вопросов сходимости этого алгоритма содержится в [125], [682]. В 1994 году был выпущен учебник, раскрывающий множество аспектов алгоритма сопряженных градиентов [981]. Доступное описание этого алгоритма в свете нейронных сетей содержится в [515].

ТАБЛИЦА 4.7. Соответствие между $f(\mathbf{x})$ и $\mathbf{E}_{av}(\mathbf{w})$

Квадратичная функция $f(\mathbf{x})$	*Функция стоимости* $\mathbf{E}_{av}(\mathbf{w})$
Вектор параметров $\mathbf{x}(n)$	Вектор синаптических весов $\mathbf{w}(n)$
Вектор градиента $\partial f(\mathbf{x})/\partial \mathbf{x}$	Вектор градиента $\partial \mathbf{E}_{av}/\partial \mathbf{w}$
Матрица $\mathbf{A}$	Матрица Гессиана $\mathbf{H}$

лось бы оценить $\beta(n)$ без точного знания матрицы $\mathbf{A}$. Такую оценку можно получить с помощью любой из следующих формул [298].

1. *Формула Полака–Рибьера* (Polak-Ribiere formula):

$$\beta(n) = -\frac{\mathbf{r}^T(n)(\mathbf{r}(n) - \mathbf{r}(n-1))}{\mathbf{r}^T(n-1)\mathbf{r}(n-1)}. \tag{4.134}$$

2. *Формула Флетчера–Ривза* (Fletcher-Reeves formula):

$$\beta(n) = -\frac{\mathbf{r}^T(n)\mathbf{r}(n)}{\mathbf{r}^T(n-1)\mathbf{r}(n-1)}. \tag{4.135}$$

Чтобы использовать метод сопряженных градиентов для решения задачи безусловной минимизации функции стоимости $\mathbf{E}_{av}(\mathbf{w})$, вытекающей из проблемы обучения многослойного персептрона, выполним две операции.

- Аппроксимируем функцию стоимости $\mathbf{E}_{av}(\mathbf{w})$ квадратичной функцией. Это значит, что третье и следующие слагаемые выражения (4.117) не будут учитываться. Таким образом, мы предполагаем работу в непосредственной близости от точки локального минимума на поверхности ошибок. Сравнивая выражения (4.117) и (4.122), можно сделать выводы, приведенные в табл. 4.7.
- Зададим формулы вычисления $\beta(n)$ и $\eta(n)$ в алгоритме сопряженных градиентов так, чтобы для определения этих параметров требовалась информация только о градиенте.

Последний пункт является особенно важным в контексте многослойного персептрона, так как он позволяет избежать вычисления матрицы Гессиана $\mathbf{H}(n)$, что сопряжено с известными вычислительными сложностями.

Для вычисления коэффициента $\beta(n)$, определяющего направление поиска $s(n)$, без точного знания матрицы Гессиана $\mathbf{H}(n)$ можно использовать формулу Полака–Рибьера (4.134) или Флетчера–Ривза (4.135). Обе эти формулы используют только уже имеющиеся данные. Для линейной формы метода сопряженных градиентов (имеется в виду квадратичная функция) эти формулы эквивалентны. С другой стороны, при неквадратичной функции стоимости эта эквивалентность теряется.

Что касается задачи неквадратичной оптимизации, форма алгоритма сопряженных градиентов на основе формулы Полака–Рибьера явно выигрывает по сравнению с модификацией алгоритма на базе формулы Флетчера–Ривза, чему можно дать эвристическое объяснение [124]. Из-за наличия в формуле функции стоимости $\mathbf{E}_{av}(\mathbf{w})$ слагаемых третьего и более высоких порядков и возможной неточности линейного поиска сопряженность генерируемых направлений поиска постепенно теряется. Это может, в свою очередь, привести к нарушению работы алгоритма (его "заеданию" или "застопориванию"), поскольку вектор направления $\mathbf{s}(n)$ и вектор $\mathbf{r}(n)$ станут приблизительно ортогональными. Если это случится, то будет выполнено равенство $\mathbf{r}(n) = \mathbf{r}(n-1)$ и, следовательно, скаляр $\beta(n)$ будет равен нулю. Тогда направление $\mathbf{s}(n)$ будет близко к вектору $\mathbf{r}(n)$, что обеспечит "прорыв" блокировки продолжения поиска. В противоположность этому при использовании формулы Флетчера–Ривза при аналогичных условиях алгоритм сопряженных градиентов обычно продолжает "заедать".

Однако в некоторых случаях метод Полака–Рибьера может зацикливаться и не сходиться. К счастью, сходимость алгоритма Полака–Рибьера можно гарантировать, приняв [981]

$$\beta = \max\{\beta_{pr}, 0\}, \tag{4.136}$$

где β_{pr} — значение, получаемое из формулы Полака–Рибьера (4.134). Использование значения β из выражения (4.136) эквивалентно повторному запуску метода сопряженных градиентов в случае $\beta<0$. Повторный запуск алгоритма эквивалентен "забыванию" последних направлений поиска и его продолжению в направлении наискорейшего спуска [981].

Теперь вернемся к следующему параметру — $\eta(n)$, определяющему скорость обучения алгоритма сопряженных градиентов. Как и в случае с $\beta(n)$, предпочтительнее использовать метод, не подразумевающий вычисления матрицы Гессиана $\mathbf{H}(n)$. Напомним, что линейная минимизация, основанная на (4.126), приводит к той же формуле для $\eta(n)$, которая была выведена из соотношения (4.125). Таким образом, требуется *линейный поиск* (line search)[16], целью которого является минимизация функции $\mathbf{E}_{av}(\mathbf{w}+\eta\mathbf{s})$ по η. При фиксированных значениях векторов $\mathbf{w}$ и $\mathbf{s}$ задача сводится к изменению параметра η с целью минимизации этой функции. По мере изменения параметра η аргумент $(\mathbf{w}+\eta\mathbf{s})$ очерчивает в W-мерном пространстве векторов w прямую линию, откуда поиск и получил название "линейного". *Алгоритм линейного поиска* (line search algorithm) является итеративной процедурой, генерирующей

[16] Стандартная форма алгоритма сопряженных градиентов требует использования линейного поиска, что из-за его характера "проб и ошибок" может занять много времени. В [749] описана модифицированная версия алгоритма сопряженных градиентов, названная алгоритмом масштабируемых сопряженных градиентов, в которой линейный поиск отсутствует. Линейный поиск был просто заменен одномерной формой алгоритма Левенберга-Маркардта (Levenberg-Marquardt). Основанием для использования именно этого метода было желание обойти сложности, вызываемые неположительной определенностью матрицы Гессиана [298].

последовательность оценок $\{\eta(n)\}$ для алгоритма сопряженных направлений. Этот алгоритм поиска останавливается по достижении удовлетворительного решения. Линейный поиск должен осуществляться в каждом из направлений поиска.

В литературе предлагается множество алгоритмов линейного поиска, и важно сделать хороший выбор, так как это оказывает первостепенное влияние на производительность алгоритма сопряженных градиентов, частью которого является линейный поиск. Алгоритмы линейного поиска включают две фазы [298].

- *Фаза группировки* (bracketing phase). В ней находится группа, представляющая собой нетривиальный интервал, гарантированно содержащий минимум.
- *Фаза разделения* (sectioning phase). В ней группа делится на подгруппы, определяющие последовательность отрезков, постепенно уменьшающихся по длине.

Опишем *процедуру подбора кривых* (curve-fitting procedure), которая объединяет в себе эти две фазы.

Пусть $\mathbf{E}_{\text{av}}(\eta)$ — функция стоимости многослойного персептрона, зависящая от параметра η. Предполагается, что функция $\mathbf{E}_{\text{av}}(\eta)$ является *унимодальной* (т.е. имеет единственный минимум в окрестности текущей точки $\mathbf{w}(n)$), дважды непрерывно дифференцируемой. Процедура начинается с поиска вдоль одной линии, пока не будут найдены три такие точки, η_1, η_2 и η_3, для которых будут выполняться следующие условия (рис. 4.25):

$$\mathbf{E}_{av}(\eta_1) \geq \mathbf{E}_{\text{av}}(\eta_3) \geq \mathbf{E}_{\text{av}}(\eta_2) \text{ для } \eta_1 < \eta_2 < \eta_3. \tag{4.137}$$

Так как $\mathbf{E}_{\text{av}}(\eta)$ является непрерывной функцией параметра η, соотношение (4.137) гарантирует, что ее минимум находится в пределах отрезка $[\eta_1, \eta_3]$. Предполагая достаточную гладкость функции $\mathbf{E}_{\text{av}}(\eta)$, можно заключить, что в окрестности точки минимума она является приблизительно параболической. Следовательно, для выполнения разделения можно использовать *обратную параболическую интерполяцию* (inverse parabolic interpolation) [858]. Эта параболическая функция должна проходить через три имеющиеся точки — η_1, η_2 и η_3 (рис. 4.26). На рис. 4.26 сплошная линия соответствует функции стоимости, а пунктирная — первой итерации процедуры разделения. Пусть минимум параболы, проходящей через точки η_1, η_2 и η_3, достигается в точке η_4. В примере, показанном на рис. 4.26, $\mathbf{E}_{\text{av}}(\eta_4) < \mathbf{E}_{\text{av}}(\eta_2)$ и $\mathbf{E}_{\text{av}}(\eta_4) < \mathbf{E}_{\text{av}}(\eta_1)$. Следовательно, точку η_3 можно заменить точкой η_4. Процедура группировки-разделения повторяется несколько раз до тех пор, пока не будет найдена точка, достаточно близкая к минимуму функции $\mathbf{E}_{\text{av}}(\eta)$. После этого линейный поиск прекращается.

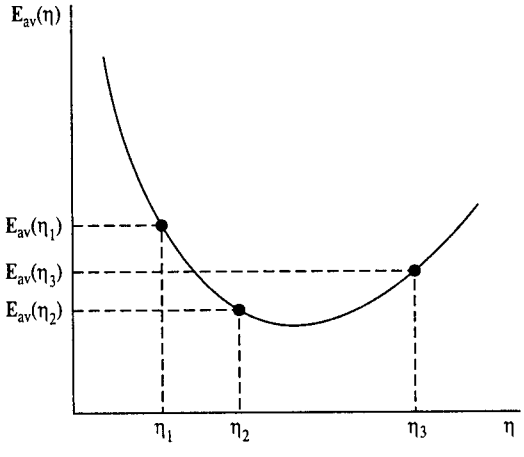


Рис. 4.25. Линейный поиск

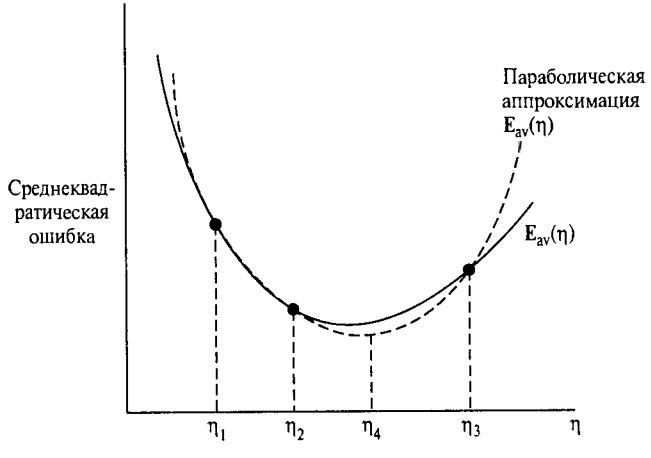


Рис. 4.26. Обратная параболическая интерполяция

Метод Брента (Brent's method) является "уточненной" версией только что продемонстрированной процедуры интерполяции по трем точкам [858]. На любом конкретном шаге вычислений метод Брента сохраняет шесть точек функции $\mathbf{E}_{av}(\eta)$. Они могут не все отличаться друг от друга. Как и ранее, параболическая интерполяция осуществляется лишь по трем из них. Чтобы такая интерполяция была приемлемой, должен выполняться соответствующий критерий, в котором участвуют и три оставшиеся точки. В конечном счете получается робастный алгоритм линейного поиска.

Нелинейный алгоритм сопряженных градиентов в сжатом виде

Теперь имеются все составляющие для формального описания нелинейной (неквадратичной) формы алгоритма сопряженных градиентов для обучения многослойного персептрона. Этот алгоритм в сжатом виде представлен в табл. 4.8.

ТАБЛИЦА 4.8. Нелинейный алгоритм сопряженных градиентов для обучения многослойного персептрона без учителя

Инициализация

Если недоступны априорные сведения о векторе весов $\mathbf{w}$, начальное значение $\mathbf{w}(0)$ выбирается с использованием процедуры, аналогичной описанной для алгоритма обратного распространения.

Вычисления

1. Для $\mathbf{w}(0)$ используем обратное распространение для вычисления вектора градиента $\mathbf{g}(0)$.
2. Устанавливаем $\mathbf{s}(0) = \mathbf{r}(0) = -\mathbf{g}(0)$.
3. На шаге n, используя линейный поиск, находим параметр $\eta(n)$, существенно минимизирующий функцию стоимости $\mathbf{E}_{\text{av}}(\eta)$, представленную как функцию от аргумента η при фиксированных значениях $\mathbf{x}$ и $\mathbf{w}$.
4. Проверяем, является ли Евклидова норма резидуальной ошибки $||\mathbf{r}(n)||$ меньше наперед заданного значения, являющегося малой частью исходного значения $||\mathbf{r}(0)||$.
5. Вычисляем вектор весов:
   $\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{s}(n)$.
6. Для $\mathbf{w}(n+1)$ используем алгоритм обратного распространения для вычисления градиента $\mathbf{g}(n+1)$.
7. Устанавливаем $\boldsymbol{r}(n+1) = -\mathbf{g}(n+1)$
8. Используем метод Полака–Рибьера для вычисления параметра $\beta(n+1)$:
   $\beta(n+1) = \max\left\{\frac{\mathbf{r}^T(n+1)(\mathbf{r}(n+1)-\mathbf{r}(n))}{\mathbf{r}^T(n)\mathbf{r}(n)}, 0\right\}$.
9. Изменяем значение вектора направления:
   $\mathbf{s}(n+1) = \mathbf{r}(n+1) + \beta(n+1)\mathbf{s}(n)$.
10. Переходим к следующей итерации $(n = n+1)$, т.е. к действию 3.

Критерий останова

Алгоритм прекращает работу, если удовлетворяется следующее условие:
$||\mathbf{r}(n)|| < \varepsilon||\mathbf{r}(0)||$,
где ε — наперед заданное малое число.

Квазиньютоновкие методы

Подводя итог обсуждению квазиньютоновских методов, можно сказать, что все они являются в сущности градиентными методами, описываемыми следующей формулой модификации весов:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{s}(n), \tag{4.138}$$

где вектор направления $\mathbf{s}(n)$ описывается в терминах вектора градиента $\mathbf{g}(n)$:

$$\mathbf{s}(n) = -\mathbf{S}(n)\mathbf{g}(n). \tag{4.139}$$

Матрица $\mathbf{S}(n)$ является положительно определенной и изменяющейся на каждой итерации. Это делается для того, чтобы вектор $\mathbf{s}(n)$ соответствовал *направлению Ньютона* (Newton direction), а именно:

$$-(\partial^2 \mathbf{E}_{\text{av}}/\partial \mathbf{w}^2)^{-1}(\partial \mathbf{E}_{\text{av}}/\partial \mathbf{w}).$$

Квазиньютоновские методы используют информацию второго порядка (о кривизне) о поверхности ошибок, не требуя точного знания матрицы Гессиана $\mathbf{H}$. Это достигается с помощью информации о значениях $\mathbf{w}(n)$ и $\mathbf{w}(n+1)$ на двух последовательных итерациях наряду с соответствующими векторами градиентов $\mathbf{g}(n)$ и $\mathbf{g}(n+1)$. Пусть

$$\mathbf{q}(n) = \mathbf{g}(n+1) - \mathbf{g}(n) \tag{4.140}$$

и

$$\Delta\mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n). \tag{4.141}$$

Тогда информацию о кривизне можно получить с помощью формулы

$$\mathbf{q}(n) \simeq \left(\frac{\partial}{\partial \mathbf{w}}\mathbf{g}(n)\right)\Delta\mathbf{w}(n). \tag{4.142}$$

В частности, имея W линейно-независимых приращений весов $\Delta\mathbf{w}(0), \Delta\mathbf{w}(1), \ldots, \Delta\mathbf{w}(W-1)$ и соответствующие приращения градиента $\mathbf{q}(0), \mathbf{q}(1), \ldots, \mathbf{q}(W-1)$, можно аппроксимировать матрицу Гессиана выражением

$$\mathbf{H} \simeq [\mathbf{q}(0), \mathbf{q}(1), \ldots, \mathbf{q}(W-1)][\Delta\mathbf{w}(0), \Delta\mathbf{w}(1), \ldots, \Delta\mathbf{w}(W-1)]^T. \tag{4.143}$$

Аналогично можно аппроксимировать матрицу, обратную Гессиану, соотношением

$$\mathbf{H}^{-1} \simeq [\Delta\mathbf{w}(0), \Delta\mathbf{w}(1), \ldots, \Delta\mathbf{w}(W-1)][\mathbf{q}(0), \mathbf{q}(1), \ldots, \mathbf{q}(W-1)]^{-1}. \tag{4.144}$$

Если функция стоимости $\mathbf{E}_{av}$ является квадратичной, равенства (4.143) и (4.144) являются точными.

В самом популярном классе квазиньютоновских методов матрица $\mathbf{S}(n+1)$ вычисляется рекурсивно с помощью предыдущего значения $\mathbf{S}(n)$ и векторов $\Delta\mathbf{w}(n)$ и $\mathbf{q}(n)$ [125], [298]:

$$\mathbf{S}(n+1) = \mathbf{S}(n) + \frac{\Delta\mathbf{w}(n)\Delta\mathbf{w}^T(n)}{\mathbf{q}^T(n)\mathbf{q}(n)} - \frac{\mathbf{S}(n)\mathbf{q}(n)\mathbf{q}^T(n)\mathbf{S}(n)}{\mathbf{q}^T(n)\mathbf{S}(n)\mathbf{q}(n)} + \\ + \xi(n)[\mathbf{q}^T(n)\mathbf{S}(n)\mathbf{q}(n)][\mathbf{v}(n)\mathbf{v}^T(n)], \tag{4.145}$$

где

$$\mathbf{v}(n) = \frac{\Delta\mathbf{w}(n)}{\Delta\mathbf{w}^T(n)\Delta\mathbf{w}(n)} - \frac{\mathbf{S}(n)\mathbf{q}(n)}{\mathbf{q}^T(n)\mathbf{S}(n)\mathbf{q}(n)} \tag{4.146}$$

и

$$0 \leq \xi(n) \leq 1 \text{ для всех } n. \tag{4.147}$$

Этот алгоритм начинается с некоторой произвольной положительно определенной матрицы $\mathbf{S}(0)$. Конкретные реализации квазиньютоновских методов отличаются выбором скаляра $\eta(n)$ [298].

- Если $\xi(n) = 0$ для всех n, получается *алгоритм Девидона–Флетчера–Пауэла* (Davidon–Fletcher–Powel — DFP), который исторически является первым квазиньютоновским методом.
- Если $\xi(n) = 1$ для всех n, получим *алгоритм Бройдена–Флетчера–Гольдфарба–Шанно* (Broyden–Fletcher–Goldfarb–Shanno), который в настоящее время считается лучшей формой квазиньютоновских методов.

Сравнение квазиньютоновских методов с методом сопряженных градиентов

Завершим этот краткий обзор квазиньютоновских методов их сравнением методом сопряженных градиентов в контексте задач неквадратичной оптимизации [125].

- Оба этих подхода (квазиньютоновский и сопряженных градиентов) избегают прямого вычисления матрицы Гессиана. Однако квазиньютоновские методы идут на шаг впереди, обобщая аппроксимацию для матрицы, обратной Гессиану. Это значит, что при точном линейном поиске и нахождении в непосредственной близости от локального минимума положительно определенного Гессиана квазиньютоновские методы стремятся аппроксимировать метод Ньютона, обеспечивая таким образом более быструю сходимость, чем это возможно при использовании метода сопряженных градиентов.
- Квазиньютоновские методы не так чувствительны к точности линейного поиска при оптимизации, как метод сопряженных градиентов.

- Квазиньютоновские методы, в дополнение к операциям матрично-векторного умножения, требуют хранения матрицы $\mathbf{S}(n)$, связанной с вычислением вектора направления $\mathbf{s}(n)$. В конечном итоге вычислительная сложность квазиньютоновских методов оценивается как $O(W^2)$. В противовес этому вычислительная сложность метода сопряженных градиентов оценивается как $O(W)$. Таким образом, в вычислительном смысле, если размерность W вектора весов $\mathbf{w}$ велика, метод сопряженных градиентов является более предпочтительным, чем квазиньютоновские методы.

Именно последний пункт ограничивает применение на практике квазиньютоновских методов только небольшими нейросетями.

4.19. Сети свертки

До сих пор рассматривались вопросы алгоритмического конструирования многослойного персептрона и его обучения. В этом разделе мы сконцентрируем внимание на структурной организации самого многослойного персептрона. В частности, будет описан особый класс многослойных персептронов, получивший название *сетей свертки* (convolutional networks). Идея, лежащая в основе этого понятия, была вкратце описана в главе 1.

Сеть свертки представляет собой многослойный персептрон, специально созданный для распознавания двумерных поверхностей с высокой степенью инвариантности к преобразованиям, масштабированию, искажениям и прочим видам деформации. Обучение решению этой сложной задачи осуществляется с учителем, при этом используются сети, архитектура которых удовлетворяет следующим ограничениям [620].

Извлечение признаков. Каждый нейрон получает входной сигнал от локального *рецептивного поля* (receptive field) в предыдущем слое, извлекая таким образом его локальные признаки. Как только признак извлечен, его точное местоположение не имеет значения, поскольку приблизительно установлено его расположение относительно других признаков.

Отображение признаков (feature mapping). Каждый вычислительный слой сети состоит из множества *карт признаков* (feature map), каждая из которых имеет форму плоскости, на которой все нейроны должны совместно использовать одно и то же множество синаптических весов. Эта форма структурных ограничений имеет следующие преимущества.

- *Инвариантность к смещению* (shift invariance), реализованную посредством карт признаков с использованием *свертки* (convolution) с ядром небольшого размера, выполняющим функцию "сплющивания".

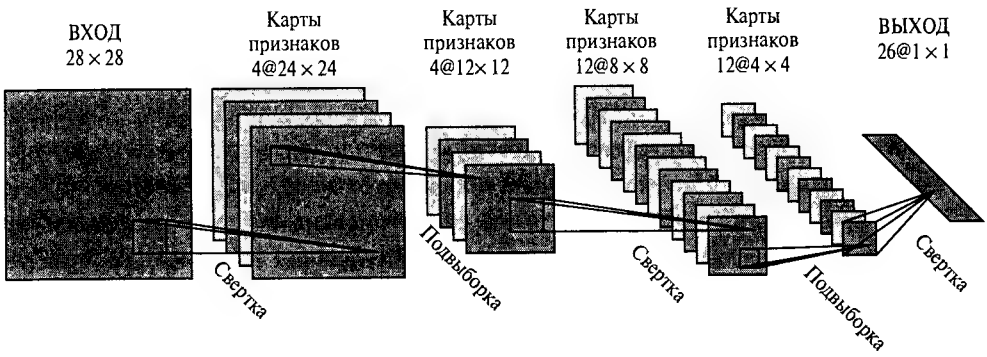


Рис. 4.27. Сеть свертки для обработки изображений, например при распознавании рукописного текста. (Воспроизведено с разрешения MIT Press.)

- *Сокращение числа свободных параметров*, реализованное с помощью совместного использования синаптических весов.

Подвыборка (subsampling). За каждым слоем свертки следует вычислительный слой, осуществляющий *локальное усреднение* (local averaging) и *подвыборку*. Посредством этого достигается уменьшение разрешения для карт признаков. Эта операция приводит к уменьшению чувствительности выходного сигнала оператора отображения признаков, к смещению и прочим формам деформации.

Такое построение сетей свертки имеет нейробиологическое обоснование, описанное в новаторских работах, посвященных локально-чувствительным и *избирательно-ориентационным* (orientation-selective) нейронам зрительного аппарата кошки [489], [490].

Следует подчеркнуть, что все веса во всех слоях сверточной сети обучаются на примерах. Более того, сеть сама учится извлекать признаки автоматически.

На рис. 4.27 показана архитектурная схема сверточной сети, состоящей из одного входного, четырех скрытых и одного выходного слоя нейронов. Эта сеть была создана для *обработки изображений* (image processing), в частности при распознавании рукописного текста. Входной слой, состоящий из матрицы 28 × 28 сенсорных узлов, получает изображения различных символов, которые предварительно смещены к центру и нормализованы по размеру. После этого вычислительные слои поочередно реализуют операции *свертки* (convolution) и подвыборки, как описывается ниже.

- Первый скрытый слой выполняет свертку. Он состоит из четырех карт признаков, каждая из которых представляет собой матрицу из 24 × 24 нейронов. Каждому нейрону соответствует поле чувствительности размером 5 × 5.
- Второй скрытый слой выполняет подвыборку и локальное усреднение. Он тоже состоит из четырех карт признаков, но теперь уже содержащих матрицы размером 12 × 12 нейронов. Каждому нейрону соответствуют рецептивное поле размером 2 × 2, настраиваемый коэффициент, настраиваемый порог и сигмоидальная функ-

ция активации. Настраиваемый коэффициент и порог определяют рабочую область нейрона. Например, при маленьком коэффициенте нейрон работает в квазилинейном режиме.

- Третий скрытый слой выполняет повторную свертку. Он состоит из 12 карт признаков, каждая из которых представляет собой матрицу из 8×8 нейронов. Каждый нейрон этого скрытого слоя может иметь синаптические связи с различными картами признаков предыдущего скрытого слоя. В противном случае его работа была бы аналогичной первому слою свертки.
- Четвертый скрытый слой осуществляет вторую подвыборку и повторное локальное усреднение. Он состоит из 12 карт признаков, однако на этот раз каждая карта содержит матрицу из 4×4 нейронов. В противном случае работа этого слоя была бы аналогична первому слою подвыборки.
- Выходной слой осуществляет последний этап свертки. Он состоит из 26 нейронов, каждому из которых соответствует одна из 26 букв латинского алфавита. Как и ранее, каждому нейрону соответствует рецептивное поле размером 4×4.

При последовательном прохождении слоев свертки и подвыборки был получен бипирамидальный эффект. Это значит, что в каждом слое свертки или подвыборки количество карт признаков увеличивается по сравнению с предыдущим при одновременном уменьшении пространственного разрешения. Идея чередования свертки и подвыборки была вызвана чередованием "простых" и "сложных" клеток[17], впервые описанным в [490]. Многослойный персептрон (см. рис. 4.27) содержит приблизительно 100000 синаптических связей, однако в то же время имеет только 2600 свободных параметров. Такое значительное сокращение количества свободных параметров было получено за счет совместного использования весов. Емкость обучаемой машины (в терминах VC-измерения), таким образом, сократилась, что, в свою очередь, повысило способность к обобщению [618]. И что еще более важно, настройка свободных параметров выполнялась с использованием стохастической (последовательной) формы обучения методом обратного распространения.

Еще одним существенным моментом явилось то, что с помощью совместного использования весов удалось реализовать сверточную сеть в параллельной форме. Это еще одно преимущество сверточных сетей по сравнению с полносвязным многослойным персептроном.

Вывод, который напрашивается в результате анализа сети, представленной на рис. 4.27, можно сформулировать так. Во-первых, многослойный персептрон управляемого размера способен обучиться сложному, многомерному и нелинейному отоб-

[17] Точка зрения Хабеля и Визеля (Habel & Viesel) на "простые" и "сложные" клетки впервые была перенесена в предметную область нейронных сетей при создании системы, получившей название *неокогнитрона* (neocognitron) [323], [326]. Однако эта машина функционировала на основе самоорганизации, в то время как сверточная сеть, показанная на рис. 4.27, обучается с учителем на базе маркированных примеров.

ражению с помощью *ограничения* (constraining) его архитектуры путем вовлечения в нее априорных знаний о поставленной задаче. Во-вторых, синаптические веса и уровни порогов могут быть обучены с помощью цикличной работы простого алгоритма обратного распространения на множестве примеров обучения.

4.20. Резюме и обсуждение

Обучение методом обратного распространения возникло как *стандартный* алгоритм обучения многослойного персептрона, по сравнению с которым все остальные алгоритмы обучения часто отбраковываются по тем или иным соображениям. Алгоритм обратного распространения получил свое название потому, что частные производные функции стоимости (как меры производительности) по свободным параметрам сети (синаптическим весам и порогам) определяются с помощью обратного распространения сигнала ошибки (вычисленного выходными нейронами) по сети, слой за слоем. Таким образом, решается задача присваивания коэффициентов доверия в самой элегантной из возможных форм. Вычислительная мощь этого алгоритма вытекает из двух его характерных особенностей.

- *Локальность* метода изменения синаптических весов и порогов в многослойном персептроне.
- *Эффективность* метода вычисления *всех* частных производных функции стоимости по свободным параметрам.

В течение каждой отдельно взятой эпохи данных обучения алгоритм обратного распространения работает в одном из двух режимов: последовательном или пакетном. В последовательном режиме все синаптические веса всех нейронов сети пересчитываются при подаче каждого примера обучения. Отсюда следует, что оценка вектора градиента поверхности ошибок, используемая во всех вычислениях, является стохастической (случайной) по своей природе. Поэтому такой режим (как, в общем, и весь метод) получил название "стохастического обратного распространения". С другой стороны, в пакетном режиме изменение синаптических весов и порогов выполняется один раз за всю эпоху. Это приводит к более точной оценке вектора градиента, используемого при вычислениях. Несмотря на свои недостатки, последовательная (стохастическая) форма обучения методом обратного распространения при создании нейронных сетей используется гораздо чаще, особенно в больших задачах. Для того чтобы добиться наилучшего результата, требуется основательная настройка алгоритма.

Особенности архитектуры многослойного персептрона зависят от области его применения. Однако хотелось бы подчеркнуть две особенности.

При распознавании образов, содержащих нелинейные объекты, *все* нейроны сети являются *нелинейными*. Эта нелинейность достигается за счет использования сигмоидальных функций, двумя самыми распространенными формами которых являются несимметричная логистическая функция и антисимметричная функция гиперболического тангенса. Каждый из нейронов отвечает за создание гиперплоскости в пространстве решений. В процессе обучения с учителем комбинация гиперплоскостей, сформированных всеми нейронами сети, итеративно трансформируется для разделения примеров, преднамеренно взятых из разных классов, а также ранее не встречавшихся, с наименьшей средней ошибкой классификации. В задачах распознавания образов при обучении наиболее широко используется алгоритм обратного распространения, особенно если размерность таких задач велика (например, при оптическом распознавании символов — OCR).

В задачах нелинейной регрессии *диапазон выходных значений* (output range) многослойного персептрона должен быть большим, чтобы вместить все данные процесса. Если эта информация недоступна, использование в выходном слое линейных нейронов будет самым правильным решением. Относительно алгоритмов обучения можно предложить следующие наблюдения.

- Последовательный (стохастический) режим обучения на основе обратного распространения является более медленным, чем пакетный.
- Пакетный режим обучения методом обратного распространения является более медленным, чем метод сопряженных градиентов. При этом следует заметить, что последний метод используется *только* в пакетном режиме.

Завершим обсуждение несколькими замечаниями относительно *меры эффективности* (performance measure). Модификации алгоритма обратного распространения, представленные в настоящей главе, основаны на минимизации функции стоимости $\mathbf{E}_{av}$, определенной как сумма квадратов ошибок, усредненная по всему множеству примеров обучения. Важной чертой этого критерия является его общность и математическая трактовка. Однако во многих ситуациях, которые встречаются на практике, минимизация функции стоимости $\mathbf{E}_{av}$ соответствует оптимизации некоторой средней величины, которая не обязательно присутствует в системе и, таким образом, может достичь только своего *субоптимального* значения. Например, в финансово-торговых системах единственной целью инвестора (равно как и маклера) является максимизация *ожидаемой прибыли* (expected return) при минимальном риске [192], [751]. Для этих задач в качестве меры производительности интуитивно больше подойдет отношение прибыли к изменчивости, чем среднеквадратическая ошибка $\mathbf{E}_{av}$.

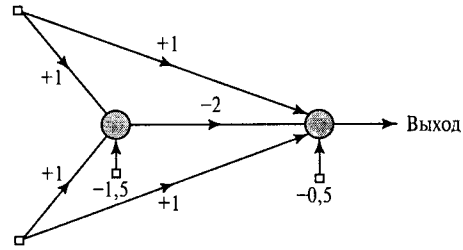


Рис. 4.28. Нейронная сеть для решения задачи XOR

Задачи

Задачи XOR

4.1. На рис. 4.28 показана нейронная сеть, содержащая один скрытый нейрон и предназначенная для решения задачи XOR. Эту сеть можно рассматривать как альтернативу предложенной в разделе 4.5. Покажите, что сеть на рис. 4.28 решает задачу XOR, построив области решений и таблицу истинности сети.

4.2. Используйте алгоритм обратного распространения для вычисления множества синаптических весов и уровней порогов в нейронной сети, показанной на рис. 4.8 и предназначенной для решения задачи XOR. Предполагается, что в качестве модели нелинейности используется логистическая функция.

Обучение методом обратного распространения

4.3. Включение слагаемого момента в формулу изменения весов можно рассматривать как механизм реализации эвристик 3 и 4, предлагающих способ ускорения сходимости алгоритма обратного распространения (см. раздел 4.17). Продемонстрируйте правильность этого утверждения.

4.4. Константа момента α обычно принимает положительное значение из диапазона $0 \leq \alpha < 1$. Проанализируйте, как изменится поведение величин, описываемых выражением (4.41), во времени t, если константе момента α присвоить отрицательное значение из диапазона $-1 < \alpha \leq 0$.

4.5. Рассмотрим простой пример сети с одним синаптическим весом, для которой задана следующая функция стоимости:

$$\mathbf{E}(w) = k_1(w - w_0)^2 + k_2,$$

где k_1, w_0 и k_2 — константы. Для минимизации этой функции используется алгоритм обратного распространения. Исследуйте влияние константы момента α на процесс обучения, вычислив количество шагов, требуемых для сходимости, в зависимости от константы α.

4.6. В разделе 4.7 было представлено качественное обоснование следующего свойства классификатора на основе многослойного персептрона (использующего в качестве модели нелинейности логистическую функцию): он обеспечивает оценку *апостериорной* вероятности принадлежности классу (a posteriori class probabilities). Это предполагает, что при достаточно большом наборе примеров обучения алгоритм обратного распространения, используемый для обучения нейронной сети, не будет останавливаться в точке локального минимума. Опишите это свойство математически.

4.7. Взяв за основу функцию стоимости из формулы (4.70), выведите минимизирующее решение, представленное формулой (4.72), и найдите минимальное значение функции стоимости, определенное в (4.73).

4.8. Равенства (4.81)–(4.83) определяют частные производные функции аппроксимации $F(\mathbf{w}, \mathbf{x})$, используемые многослойным персептроном на рис. 4.18. Выведите эти равенства, используя следующий сценарий:

а) *Функция стоимости:*

$$\mathbf{E}(n) = \frac{1}{2}[d - F(\mathbf{w}, \mathbf{x})]^2.$$

б) *Выход нейрона j:*

$$y_j = \varphi\left(\sum_i w_{ji} y_i\right),$$

где w_{ji} — синаптические веса связей между нейронами j и i; y_i — выходной сигнал нейрона i.

в) *Нелинейность:*

$$\varphi(v) = \frac{1}{1 + \exp(-v)}.$$

Перекрестная проверка

4.9. Можно доказать, что перекрестная проверка является частным случаем минимизации структурного риска, который рассматривался в главе 2. Приведите пример нейронной сети, использующей перекрестную проверку, который подтверждает это утверждение.

4.10. При многократной перекрестной проверке не существует четкого разделения между данными для оценивания и тестирования. Можно ли использовать многократную перекрестную проверку для нахождения *пороговой оценки* (biased estimate)? Обоснуйте свой ответ.

Приемы упрощения сети

4.11. Статистические критерии выбора модели, такие как *критерий минимальной длины описания Риссанена* (Rissanen's minimum description length criterion) и *информационно-теоретический критерий Акейка* (Information-theoretic criterion due to Akaike), имеют сходную форму построения:

$$\begin{pmatrix}\text{Критерий}\\ \text{сложности}\\ \text{модели}\end{pmatrix} = \begin{pmatrix}\text{Логарифмическая}\\ \text{функция}\\ \text{подобия}\end{pmatrix} + \begin{pmatrix}\text{Штраф}\\ \text{за сложность}\\ \text{модели}\end{pmatrix}$$

Обоснуйте, как использование методов снижения и исключения весов связей вписывается в такой формализм.

4.12. а) Выведите формулу вычисления значения выпуклости S_i (4.105).

б) Предполагая, что матрицу Гессиана среднеквадратической ошибки многослойного персептрона можно аппроксимировать следующей диагональной матрицей:

$$\mathbf{H} = \text{diag}[h_{11}, h_{22}, \ldots, h_{WW}],$$

где W — общее число весов сети, определите значение выпуклости S_i для веса w_i сети.

Ускорение сходимости алгоритма обратного распространения

4.13. Правило обучения *delta-bar-delta* [505] представляет собой видоизмененную форму алгоритма, построенного на эвристике, описанной в разделе 4.17. В этом правиле каждому синаптическому весу сети соответствует собственный параметр скорости обучения. Поэтому функция стоимости соответствующим образом изменяет вид. Другими словами, несмотря на математическую схожесть с обычной функцией стоимости, параметрическое пространство этой новой функции стоимости включает в себя другие параметры интенсивности обучения.

а) Выведите формулу для частной производной $\partial E(n)/\partial \eta_{ji}(n)$, где $\eta_{ji}(n)$ – параметр скорости обучения, связанный с синаптическим весом $w_{ji}(n)$.

б) Продемонстрируйте, что изменение параметров скорости обучения находится в полном соответствии с эвристиками 3 и 4 из раздела 4.17.

Методы оптимизации второго порядка

4.14. Использование слагаемого момента в формуле корректировки весов (4.39) можно рассматривать как аппроксимацию метода сопряженных градиентов [103]. Обоснуйте корректность этого утверждения.

4.15. Начиная с формулы (4.133) для $\beta(n)$, выведите *формулу Хестенесса–Штифеля* (Hesteness–Stiefel):

$$\beta(n) = \frac{\mathbf{r}^T(n)(\mathbf{r}(n) - \mathbf{r}(n-1))}{\mathbf{S}^T(n-1)\mathbf{r}(n-1)},$$

где $\mathbf{s}(n)$ — вектор направления; $\mathbf{r}(n)$ — резидуальная ошибка метода сопряженных градиентов. Используйте этот результат для вывода формул Полака–Рибьера (4.134) и Флетчера–Ривза (4.135).

Компьютерное моделирование

4.16. Исследуйте применимость обучения методом обратного распространения с сигмоидальной нелинейностью для построения следующих отображений (скалярных):

а) $f(x) = \frac{1}{x} \quad 1 \leq x \leq 100$;

б) $f(x) = \lg_{10} x, 1 \leq x \leq 10$;

в) $f(x) = \exp(-x), 1 \leq x \leq 10$;

г) $f(x) = \sin(x), 1 \leq x \leq \pi/2$.

Для каждого отображения выполните следующее.

а) Создайте два набора данных: один — для обучения, а второй — для тестирования.

б) Используйте множество данных тестирования для вычисления синаптических весов сети, предполагая наличие одного скрытого слоя.

в) Оцените вычислительную сложность сети с помощью данных тестирования.

Используя единственный скрытый слой с различным количеством нейроном, исследуйте, как изменение количества нейронов скрытого слоя влияет на производительность сети.

4.17. Данные, приведенные в табл. 4.9, представляют вес глазных линз дикого австралийского кролика как функции времени (возраста). Ни одна простая аналитическая функция не может точно интерполировать эти данные, так как неизвестен общий вид самой функции. Поэтому введем нелинейную модель этого множества данных на основе метода наименьших квадратов, использующую отрицательный экспоненциал:

$$y = 233.846(1 - \exp(-0.006042x)) + \varepsilon,$$

где ε — слагаемое ошибки.

Используя алгоритм обратного распространения, постройте многослойный персептрон, реализующий аппроксимацию этого множества данных на основе метода наименьших квадратов. Сравните результат с описанной выше моделью.

ТАБЛИЦА 4.9. Вес глазных линз дикого австралийского кролика

Возраст (дни)	*Вес (мг)*	*Возраст (дни)*	*Вес (мг)*	*Возраст (дни)*	*Вес (мг)*	*Возраст (дни)*	*Вес (мг)*
15	21,66	75	94,6	218	174,18	338	203,23
15	22,75	82	92,5	218	173,03	347	188,38
15	22,3	85	105	219	173,54	354	189,7
18	31,25	91	101,7	224	178,86	357	195,31
28	44,79	91	102,9	225	177,68	375	202,63
29	40,55	97	110	227	173,73	394	224,82
37	50,25	98	104,3	232	159,98	513	203,3
37	46,88	125	134,9	232	161,29	535	209,7
44	52,03	142	130,68	237	187,07	554	233,9
50	63,47	142	140,58	246	176,13	591	234,7
50	61,13	147	155,3	258	183,4	648	244,3
60	81	147	152,2	276	186,26	660	231
61	73,09	150	144,5	285	189,66	705	242,4
64	79,09	159	142,15	300	186,09	723	230,77
65	79,51	165	139,81	301	186,7	756	242,57
65	65,31	183	153,22	305	186,8	768	232,12
72	71,9	192	145,72	312	195,1	860	246,7
75	86,1	195	161,1	317	216,41		

Сети на основе радиальных базисных функций

5.1. Введение

Процесс создания нейронных сетей, обучаемых с учителем, можно упростить множеством способов. Описанный в предыдущей главе алгоритм обратного распространения для многослойного персептрона можно рассматривать как реализацию рекурсивной технологии, которая в статистике называется *стохастической аппроксимацией* (stochastic approximation). В этой главе описывается еще один подход, в рамках которого построение нейронной сети рассматривается как *задача аппроксимации кривой по точкам* (curve-fitting problem) в пространстве высокой размерности. В соответствии с такой точкой зрения обучение эквивалентно нахождению такой поверхности в многомерном пространстве, которая наиболее точно соответствует данным обучения. При этом критерий "наилучшего соответствия" выбирается в некотором статистическом смысле. Таким образом, обобщение эквивалентно использованию этой многомерной поверхности для интерполяции данных тестирования. Такой подход лежит в основе метода радиальных базисных функций, состоящего в традиционной интерполяции в многомерном пространстве. В контексте нейронных сетей скрытые нейроны реализуют набор "функций", являющихся произвольным "базисом" для разложения входных образов (векторов). Соответствующие преобразования называют *радиальными базисными функциями* (radial-basis function)[1]. Понятие радиальных базисных функций впервые было введено при решении задачи интерполяции вещественных функций нескольких переменных. Анализ ранних работ по этой тематике представлен в [855], а более новых — в [642]. В настоящее время радиальные базисные функции составляют одно из главных направлений исследований в области численного анализа.

[1] Радиальные базисные функции впервые использовались при решении задачи многомерной интерполяции. Анализ первых работ по этой тематике содержится в [855]. В настоящее время это одно из главных направлений исследований в области численного анализа.

В [160] радиальные базисные функции впервые использовались для построения нейронных сетей. Важный вклад в теорию и методологию проектирования сетей на основе радиальных базисных функций внесла работа, в которой основное внимание уделяется вопросам применения теории регуляризации к этому классу сетей с целью повышения качества обобщения на новых данных [847].

Базовая архитектура *сетей на основе радиальных базисных функций* (radial-basis function network — RBF), или RBF-сетей, предполагает наличие трех слоев, выполняющих совершенно различные функции. Входной слой состоит из сенсорных элементов, которые связывают сеть с внешней средой. Второй слой является единственным *скрытым* (hidden) слоем сети. Он выполняет нелинейное преобразование входного пространства в скрытое. В большинстве реализаций скрытое пространство имеет более высокую размерность, чем входное. Математическое обоснование целесообразности последовательного применения нелинейного и линейного преобразований приведено в [219]. Согласно этой работе, задача классификации данных в пространстве более высокой размерности с большей вероятностью удовлетворяет требованию линейной разделимости. Поэтому в RBF-сетях размерность скрытого слоя, как правило, существенно превышает размерность входного слоя. Также важно отметить тот факт, что размерность скрытого пространства непосредственно связана со способностью сети аппроксимировать гладкое отображение "вход-выход" [731], [787]. Чем выше размерность скрытого слоя, тем более высокой будет точность аппроксимации.

Структура главы

Эта глава организована следующим образом. В разделах 5.2 и 5.4 будут заложены основы построения RBF-сетей. Это будет сделано в два этапа. В первую очередь будет описана теорема Ковера (Cover) о разделимости образов. Для иллюстрации применения этой теоремы мы рассмотрим задачу XOR. Раздел 5.3 посвящен задаче интерполяции и ее связи с сетями RBF.

После изучения основ функционирования RBF-сетей мы перейдем ко второй части главы, состоящей из разделов 5.4–5.9. В разделе 5.4 будет показано, что обучение с учителем является плохо обусловленной задачей восстановления гиперповерхности. В разделе 5.5 детально описывается теория регуляризации Тихонова и рассматривается ее применение в сетях RBF. Эта теория естественным образом приводит к формулировке понятия сетей регуляризации в разделе 5.6. Этот класс RBF-сетей является очень требовательным к вычислительным ресурсам. Чтобы уменьшить эту сложность, в разделе 5.7 описываются специальные сети регуляризации, которые называются обобщенными RBF-сетями. В разделе 5.9 мы снова вернемся к задаче XOR и покажем, как ее можно решить с помощью RBF-сетей. В разделе 5.9 изучение теории регуляризации будет завершено описанием метода обобщенной перекрестной проверки для выбора подходящего параметра регуляризации.

В разделе 5.10 обсуждаются свойства аппроксимации RBF-сетей. В разделе 5.11 проводится сравнительный анализ сетей на основе радиальных базисных функций и многослойных персептронов. Оба этих типа сетей являются важными примерами многослойных сетей прямого распространения.

В разделе 5.12 предлагается еще один подход к изучению RBF-сетей — с позиций оценивания регрессии ядра. Здесь область RBF-сетей связывается с вопросами оценки плотности и теорией регрессии ядра.

В последней части настоящей главы (разделы 5.13 и 5.14) описываются четыре различные стратегии обучения нейронных сетей на основе радиальных базисных функций (раздел 5.13), а также компьютерный эксперимент по решению задачи классификации образов с использованием RBF-сетей (раздел 5.14).

Глава завершается несколькими заключительными рассуждениями о RBF-сетях (раздел 5.15).

5.2. Теорема Ковера о разделимости множеств

Если сеть на основе радиальных базисных функций (RBF-сеть) используется для решения сложных задач классификации образов, то основная идея решения обычно состоит в нелинейном преобразовании входных данных в пространстве более высокой размерности. Теоретическую основу такого подхода составляет *теорема Ковера о разделимости образов* (Cover's theorem on the separability of patterns), которая утверждает следующее [219].

> *Нелинейное преобразование сложной задачи классификации образов в пространство более высокой размерности повышает вероятность линейной разделимости образов.*

Из главы 3, посвященной однослойному персептрону, известно, что задача классификации линейно-разделимых множеств относительно легко разрешима. Следовательно, для более четкого понимания принципов работы RBF-сети в качестве классификатора необходимо глубже изучить вопрос *разделимости образов* (separability of patterns).

Рассмотрим семейство поверхностей, каждая из которых делит входное пространство на две части. Пусть $\mathbf{X}$ — множество, состоящее из N образов (векторов) $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, каждый из которых принадлежит одному из двух классов — $\mathbf{X}_1$ или $\mathbf{X}_2$. Эта *дихотомия* (бинарное разбиение) точек называется разделимой по отношению к семейству поверхностей, если в этом семействе существует поверхность, которая отделяет точки класса $\mathbf{X}_1$ от точек класса $\mathbf{X}_2$. Для каждого образа $\mathbf{x} \in \mathbf{X}$ определим вектор, состоящий из множества действительнозначных функций $\{\varphi_i(\mathbf{x}) | \quad i = 1, 2, \ldots, m_1\}$, вида

$$\varphi(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T. \tag{5.1}$$

Предположим, что образ $\mathbf{x}$ является вектором в m_0-мерном входном пространстве. Тогда векторная функция $\boldsymbol{\varphi}(\mathbf{x})$ отображает точки m_0-мерного входного пространства в новое пространство размерности m_1. Функции $\varphi_i(\mathbf{x})$ называются *скрытыми*, поскольку они играют роль скрытых элементов нейронных сетей прямого распространения. Соответственно пространство, образованное множеством скрытых функций $\{\varphi_i(\mathbf{x})\}_{i=1}^{m_1}$, называется *скрытым пространством* (hidden space) или *пространством признаков* (feature space).

Дихотомия $\{\mathbf{X}_1, \mathbf{X}_2\}$ множества $\mathbf{X}$ называется φ-*разделимой*(φ-separable), если существует m_1-мерный вектор $\mathbf{w}$, для которого можно записать [219]

$$\begin{aligned} \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) > 0,\ \mathbf{x} \in \mathbf{X}_1, \\ \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) < 0,\ \mathbf{x} \in \mathbf{X}_2. \end{aligned} \tag{5.2}$$

Гиперплоскость, задаваемая уравнением

$$\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) = 0,$$

описывает поверхность в φ-пространстве (т.е. в скрытом пространстве). Обратный образ этой поверхности, т.е.

$$\mathbf{x} : \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) = 0, \tag{5.3}$$

определяет *разделяющую поверхность* (separating surface) во входном пространстве.

Рассмотрим естественный класс отображений, получаемый при использовании линейной комбинации произведений r координат вектора входного образа. Разделяющие поверхности, соответствующие такому отображению, называются *рациональными многообразиями r-го порядка* (r-th order rational variety). Рациональное многообразие r-го порядка в пространстве размерности m_0 описывается однородным уравнением r-го порядка в координатах входного вектора $\mathbf{x}$:

$$\sum_{0 \le i_1 \le i_2 \le \ldots \le i_r \le m_0} a_{i_1 i_2 \ldots i_r} x_{i_1} x_{i_2} \ldots x_{i_r} = 0, \tag{5.4}$$

где x_i — i-й компонент входного вектора $\mathbf{x}$, причем x_0 присвоено значение 1, чтобы придать уравнению однородную форму. Произведение элементов x_i вектора $\mathbf{x}$ r-го порядка (т.е. $x_{i1}x_{i2}\ldots x_{ir}$) называется *одночленом* (monomial). Для входного пространства размерности m_0 сумма в (5.4) включает

$$\frac{(m_0 - r)!}{m_0! r!}$$

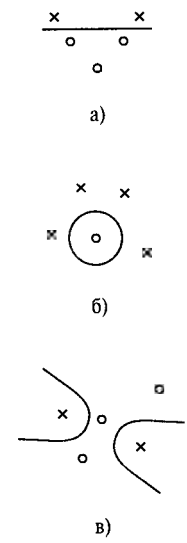


Рис. 5.1. Три примера φ-разделимых дихотомий для различных множеств из пяти точек в двумерном пространстве: линейно-разделимая дихотомия (а); сферически разделимая дихотомия (б); квадратично-разделимая дихотомия (в)

одночленов. Примерами разделяющих поверхностей, описываемых уравнением (5.4), являются *гиперплоскости* (hyperplane) (рациональные многообразия первого порядка), *квадрики* (quadrice) (рациональные многообразия 2-го порядка) и *гиперсферы* (hypersphere) (квадрики с некоторыми линейными ограничениями на коэффициенты). Эти примеры показаны на рис. 5.1 для конфигурации из пяти точек в двумерном входном пространстве. В общем случае линейная разделимость предполагает сферическую разделимость, которая, в свою очередь, предполагает квадратичный вид классифицирующей функции. Обратное утверждение не всегда верно.

В вероятностном эксперименте разделимость множества образов становится случайным событием, зависящим от выбранной дихотомии и распределения образов во входном пространстве. Предположим, что образы активации $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ выбираются независимо, в соответствии с вероятностным распределением, присущим входному пространству. Предположим также, что все возможные дихотомии $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ равновероятны. Пусть $P(N, m_1)$ — вероятность того, что некоторая случайно выбранная дихотомия является φ-разделимой, если класс разделяющих гиперповерхностей имеет m_1 степеней свободы. Согласно [219] можно утверждать, что

$$P(N, m_1) = \left(\frac{1}{2}\right)^{N-1} \sum_{m=0}^{m_1-1} \binom{N-1}{m}, \tag{5.5}$$

где биномиальные коэффициенты, включающие $N-1$ и m, для всех целых l и m определяются следующей формулой:

$$\binom{l}{m} = \frac{l(l-1)\ldots(l-m+1)}{m!}.$$

Уравнение (5.5) отражает сущность *теоремы Ковера о разделимости* (Cover's separability theorem) случайных образов[2]. Это соотношение описывает совокупное биномиальное распределение, соответствующее вероятности того, что $(N-1)$ подбрасываний монетки приведет к выпадению не более $(m_1 - 1)$ решек.

Хотя поверхности скрытых элементов в уравнении (5.5) имеют полиномиальную форму и, следовательно, отличаются от тех, которые обычно используются в сетях на основе радиальных базисных функций, сущность этого выражения носит общий характер. В частности, чем выше размерность m_1 скрытого пространства, тем ближе вероятность $P(N, m_1)$ к единице. Подводя итог сказанному, следует отметить, что теорема Ковера о разделимости образов базируется на двух основных моментах.

1. Определение нелинейной скрытой функции $\varphi_i(\mathbf{x})$, где $\mathbf{x}$ — входной вектор, а $i = 1, 2, \ldots, m_1$.
2. Высокая размерность скрытого пространства по сравнению с размерностью входного. Эта размерность определяется значением, присваиваемым m_1(т.е. количеством скрытых нейронов).

Как утверждалось ранее, в общем случае преобразование сложной задачи классификации в пространство более высокой размерности повышает вероятность линейной разделимости образов. Однако хочется подчеркнуть, что в некоторых случаях для обеспечения линейной разделимости достаточно нелинейного преобразования (п. 1) без повышения размерности пространства скрытых элементов. Это будет продемонстрировано на следующем примере.

---

[2] Доказательство теоремы Ковера базируется на следующих утверждениях [219].

- *Теорема Шлафли* (Schlafli's theorem), или *теорема подсчета функций* (function-counting theorem), которая утверждает, что количество однородных линейно-разделимых дихотомий N векторов Евклидова m_1-мерного пространства, находящихся в стандартном положении, определяется соотношением

$$C(n, m_1) = 2\sum_{m=0}^{m_1-1}\binom{N-1}{m}.$$

Говорят, что множество векторов $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ Евклидова m_1-мерного пространства находится в *стандартном положении*, если любое подмножество из m_1 или меньшего числа векторов является линейно-независимым.

- *Рефлексивная инвариантность* совместной вероятности распределения $\mathbf{X}$, которая означает, что вероятность (условная по $\mathbf{X}$) линейной разделимости случайной дихотомии равна безусловной вероятности того, что любая конкретная дихотомия множества $\mathbf{X}$ (все N векторов находятся в одной категории) будет разделимой.

Теорема подсчета функций была независимо доказана в различных формах и применена к специальной конфигурации персептронов (т.е. линейных пороговых элементов) в [168], [528], [1162]. В [218] эта теорема использовалась для оценки емкости сети персептронов в терминах общего количества настраиваемых параметров. В статье было показано, что эта емкость ограничена снизу величиной $N/(1+\log_2 N)$, где N — количество входных образов.

ТАБЛИЦА 5.1. Значения скрытых функций в задаче XOR (см. пример 5.1)

Входной образ, $\mathbf{x}$	*Первая скрытая функция*, $\varphi_1(\mathbf{x})$	*Вторая скрытая функция*, $\varphi_2(\mathbf{x})$
(1;1)	1	0,1353
(0;1)	0,3678	0,3678
(0;0)	0,1353	1
(1;0)	0,3678	0,3678

Пример 5.1

Задача XOR

Чтобы проиллюстрировать значимость идеи φ-разделимости образов, рассмотрим простую, но в то же время важную задачу XOR. В этой задаче в двумерном входном пространстве рассматриваются четыре точки: (0; 0), (0; 1), (1; 0) и (1; 1) (рис. 5.2, *а*). Требуется построить такой классификатор, который относит образы (0; 1) и (1; 0) к классу 1, а образы (0; 0) и (1; 1) — к классу 0. Это значит, что ближайшие друг к другу точки во входном пространстве (в смысле *расстояния Хемминга* (Hamming distance)) отображаются в области, которые максимально удалены друг от друга в выходном пространстве.

Определим пару гауссовых функций следующим образом:

$$\varphi_1(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{t}_1\|^2}, \quad \mathbf{t}_1 = [1, \ 1]^T,$$
$$\varphi_2(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{t}_2\|^2}, \quad \mathbf{t}_2 = [0, \ 0]^T.$$

Проанализируем представленный в табл. 5.1 результат для четырех различных входных образов. Входные векторы отображаются на плоскость $\varphi_1 - \varphi_2$, как показано на рис. 5.2, *б*. На этом рисунке сразу видно, что теперь точки (0; 1) и (1; 0) стали линейно-отделимыми от точек (0; 0) и (1; 1). Следовательно, задачу XOR можно успешно решить с помощью функций $\varphi_1(\mathbf{x})$ и $\varphi_2(\mathbf{x})$ для предварительной обработки входных данных линейного классификатора, такого как персептрон. ■

В этом примере не повышается размерность скрытого пространства по сравнению с входным. Другими словами, нелинейного преобразования, представленного в данном случае двумя гауссовыми функциями, оказалось достаточно для трансформации задачи XOR в линейно-разделимую задачу.

Разделяющая способность поверхности

Уравнение (5.5) позволяет оценить максимальное количество случайно выбранных образов, линейно-разделимых в многомерном пространстве. Чтобы раскрыть этот вопрос, рассмотрим последовательность случайных векторов $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. Пусть N — случайная переменная, определяемая как максимальное целое число, для которого эта последовательность является φ-разделимой, если φ имеет m_1 степеней

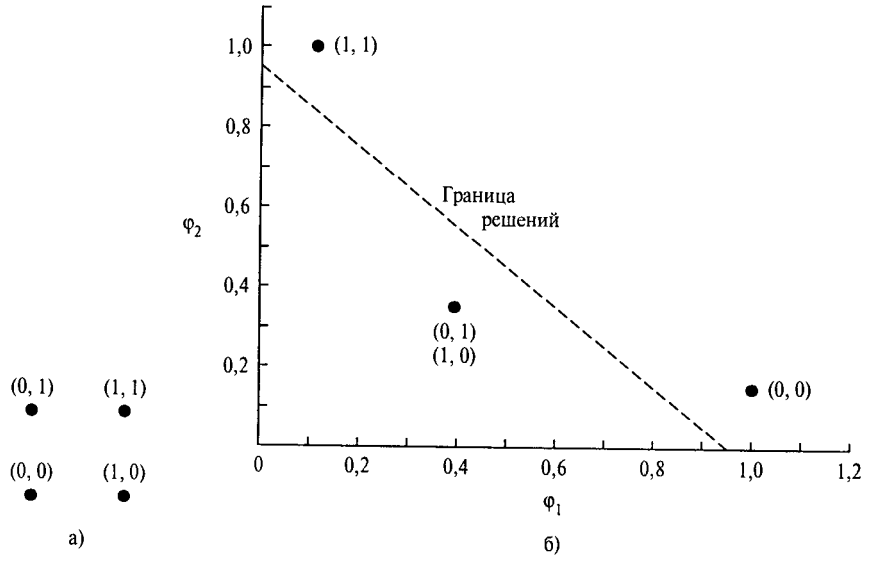


Рис. 5.2. Четыре образа для задачи XOR (а) и диаграмма принятия решений (б)

свободы. Тогда из соотношения (5.5) можно получить вероятность того, что $N = n$:

$$P(N = n) = P(n, m_1) - P(n+1, m_1) = \\ = \left(\frac{1}{2}\right)^n \binom{n-1}{m_1 - 1}, \quad n = 0,\ 1,\ 2, \ldots \quad . \tag{5.6}$$

Для интерпретации этого результата вспомним определение *отрицательного биномиального распределения* (negative binomial distribution). Это распределение определяет вероятность того, что r-му успешному эксперименту предшествует k ошибок в длинной повторяющейся последовательности *попыток Бернулли* (Bernoulli trial). В таком вероятностном эксперименте каждый результат может принимать одно из двух значений — успех или ошибка, — и их вероятности одинаковы на протяжении всего эксперимента. Пусть p и q — соответственно вероятности успеха и неудачи и $p + q = 1$. Отрицательное биномиальное распределение имеет вид [294]

$$f(k; r, p) = p^r q^k \binom{r + k - 1}{k}.$$

Для частного случая, когда $p = q = 1/2$ (т.е. успех и ошибка равновероятны) и $k + r = n$, отрицательное биномиальное распределение сводится к виду

$$f(k; n - k, \frac{1}{2}) = \left(\frac{1}{2}\right)^n \binom{n-1}{k}, \quad n = 0,\ 1,\ 2, \ldots .$$

Учитывая это определение, несложно заметить, что результат, описанный выражением (5.6), является всего лишь отрицательным биномиальным распределением, смещенным на m_1 единиц вправо, с параметрами m_1 и 1/2. Таким образом, N соответствует "времени ожидания" m_1-й ошибки в последовательности попыток подбрасывания монетки. Ожидание случайной переменной N и ее медиана (median) соответственно равны

$$E[N] = 2m_1, \tag{5.7}$$

$$\text{Median}[N] = 2m_1. \tag{5.8}$$

Таким образом, мы получили следствие из теоремы Ковера в форме асимптотического результата, который можно сформулировать следующим образом.

Ожидаемое максимальное количество случайно задаваемых образов (векторов), линейно-разделимых в пространстве размерности m_1, составляет $2m_1$.

Полученный результат подтверждает, что $2m_1$ является совершенно естественным определением *разделяющей способности* (separating capacity) семейства поверхностей решений, имеющих m_1 степеней свободы. Следовательно, разделяющая способность поверхности тесно связана со значением VC-измерения, которое рассматривалось в главе 2.

5.3. Задача интерполяции

Важным выводом, следующим из теоремы Ковера о разделимости образов, является то, что при решении нелинейной задачи классификации обычно имеется практическая польза от преобразования входного пространства в пространство более высокой размерности. В основном для трансформации нелинейной задачи классификации в линейную применяется нелинейное отображение. Аналогично, нелинейное отображение можно использовать для преобразования сложной задачи нелинейной фильтрации в более простую задачу линейной фильтрации.

Рассмотрим сеть прямого распространения с одним входным, одним скрытым и выходным слоем, содержащим единственный нейрон. Один нейрон в выходном слое выбран специально для упрощения выкладок без потери общности. Эта сеть предназначена для *нелинейного отображения* входного пространства в скрытое, за которым следует *линейное отображение* скрытого пространства в выходное. Пусть m_0 — размерность входного пространства. Тогда сеть в целом реализует отображение m_0-мерного входного пространства в одномерное выходное:

$$s : \Re^{m_0} \rightarrow \Re^1. \tag{5.9}$$

Отображение s можно рассматривать как гиперповерхность (график) $\Gamma \subset \Re^{m_0+1}$, аналогично тому, как отображение $s{:}\Re^1 \rightarrow \Re^1$ представляется в двумерном пространстве $\Re^2$ в виде параболы $s(x) = x^2$. Поверхность Г является многомерным графиком изменения выходного сигнала в зависимости от входного. В практической реализации поверхность Г остается неизвестной, а на данные обучения накладывается шум. Этапы обучения и обобщения можно представить следующим образом [160].

- На *этапе обучения* (training phase) поверхность Г оптимизируется на основании известных точек данных, представляемых сети в форме маркированных примеров типа "вход-выход".
- *Фаза обобщения* (generalization phase) равносильна интерполяции на интервалах между точками данных. Эта интерполяция осуществляется на ограниченной поверхности, сгенерированной *процедурой подбора* (fitting procedure) в качестве оптимальной аппроксимации истинной поверхности Г.

Таким образом, мы приходим к теории *интерполяции функции многих переменных* (multivariable interpolation) в многомерном пространстве, которая имеет свою долгую историю [243]. Задачу интерполяции в ее изначальном смысле можно сформулировать следующим образом.

Для данного множества из N точек $\{\boldsymbol{x}_i \in \Re^{m_0} | i = 1, 2, \ldots, N\}$ *и соответствующего множества из N действительных чисел* $\{d_i \in \Re^1 | i = 1, 2, \ldots, N\}$ *найти функцию* $F : \Re^N \rightarrow \Re^1$, *удовлетворяющую следующему условию интерполяции:*

$$F(\mathbf{x}_i) = d_i, i = 1, 2, \ldots, N. \tag{5.10}$$

Для определенной таким образом задачи поверхность интерполяции (т.е. функция F) проходит через *все* точки примеров обучения.

Метод *радиальных базисных функций* (RBF) сводится к выбору функции F, имеющей следующий вид [854]:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi(\|\mathbf{x} - \mathbf{x}_i\|), \tag{5.11}$$

где $\{\varphi(||\mathbf{x} - \mathbf{x}_i||) | i = 1, 2, \ldots, N\}$ — множество из N произвольных (и обычно нелинейных) функций, которые называются *радиальными базисными функциями* (radial-basis function); $||\cdot||$ — норма, обычно Евклидова. Известные точки данных $\mathbf{x}_i \in \Re^{m_0}, i = 1, 2, \ldots, N$, выбираются в качестве центров радиальных базисных функций.

Подставляя в (5.11) условие интерполяции (5.10), получим следующую систему линейных уравнений для неизвестных весовых коэффициентов $\{w_i\}$:

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_{N1} & \varphi_{N2} & \cdots & \varphi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdots \\ d_N \end{bmatrix}, \tag{5.12}$$

где

$$\varphi_{ji} = \varphi(||\mathbf{x}_j - \mathbf{x}_i||), (i, j) = 1, 2, \ldots, N. \tag{5.13}$$

Пусть

$$\mathbf{d} = [d_1, d_2, \ldots, d_N]^T,$$
$$\mathbf{w} = [w_1, w_2, \ldots, w_N]^T.$$

Векторы $\mathbf{d}$ и $\mathbf{w}$ размерности N — это *вектор желаемого отклика* (desired response vector) и *вектор весов* (weight vector) соответственно. Пусть $\mathbf{\Phi}$ — матрица размерности $N \times N$ с элементами φ_{ji}:

$$\mathbf{\Phi} = \{\varphi_{ji} | (j, i) = 1, 2, \ldots, N\}. \tag{5.14}$$

Назовем ее *матрицей интерполяции*. Теперь выражение (5.12) можно переписать в следующем виде:

$$\mathbf{\Phi w} = \mathbf{x}. \tag{5.15}$$

Предполагая, что матрица $\mathbf{\Phi}$ является несингулярной (и, следовательно, для нее существует обратная матрица $\mathbf{\Phi}^{-1}$), можно приступить к решению уравнения (5.15) относительно вектора весов $\mathbf{w}$:

$$\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{x}. \tag{5.16}$$

При этом возникает жизненно важный вопрос: как убедиться в несингулярности матрицы $\mathbf{\Phi}$? Для большого класса радиальных базисных функций при определенных условиях ответ на этот вопрос дает следующая важная теорема.

Теорема Мичелли

В [733] была доказана следующая теорема.

Пусть $\{\mathbf{x}_i\}_{i=1}^N$ *— множество различных точек из* $\Re^{m_0}$. *Тогда матрица интерполяции* $\mathbf{\Phi}$ *размерности* $N \times N$ *с элементами* $\varphi_{ji} = \varphi(||\mathbf{x}_j - \mathbf{x}_i||)$ *является несингулярной.*

Теорема Мичелли охватывает широкий класс радиальных базисных функций. В этот класс входят следующие функции, представляющие интерес при изучении сетей на основе радиальных базисных функций, или сетей RBF.

1. *Мультиквадратичная функция* (multiquadric):

$$\varphi(r) = (r^2 + c^2)^{1/2} \text{ для некоторых } c > 0 \text{ и } r \in \Re. \tag{5.17}$$

2. *Обратная мультиквадратичная функция* (inverse multiquadric):

$$\varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \text{ для некоторых } c > 0 \text{ и } r \in \Re. \tag{5.18}$$

3. *Функция Гаусса* (Gaussian function):

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \text{ для некоторых } \sigma > 0 \text{ и } r \in \Re. \tag{5.19}$$

Выражения для прямой и обратной мультиквадратичных функций даны согласно [419].

Для того чтобы радиальные базисные функции (5.17)-(5.19) были несингулярными, все точки $\{\mathbf{x}_i\}_{i=1}^N$ должны различаться. Больше для несингулярности матрицы $\mathbf{\Phi}$ ничего не требуется, кроме размерности N множества примеров и размерности m_0 векторов $\mathbf{x}_i$.

Обратная мультиквадратичная функция, представленная формулой (5.18), и функция Гаусса (5.19) имеют одно общее свойство: они являются *локализованными* (localized) в том смысле, что $\varphi(r) \to 0$ при $r \to \infty$. В обоих этих случаях матрица $\mathbf{\Phi}$ является положительно определенной. В отличие от них мультиквадратичная функция (5.17) не является локализованной в том смысле, что при $r \to \infty$ функция неограниченно возрастает. Соответствующая ей матрица $\mathbf{\Phi}$ имеет $N - 1$ отрицательных собственных чисел и только одно положительное, следовательно, не является положительно определенной [733]. Однако примечательно, что матрица интерполяции $\mathbf{\Phi}$, основанная на мультиквадратичной функции Харди, является несингулярной и, таким образом, пригодной для использования в конструкции сетей RBF.

Еще более важным является тот факт, что радиальные базисные функции, неограниченно возрастающие при стремлении аргумента к бесконечности (например, мультиквадратичные функции), можно использовать для аппроксимации гладких отображений с большей точностью, чем при использовании положительно-определенной матрицы. Этот удивительный результат рассматривается в [854].

5.4. Обучение с учителем как плохо обусловленная задача восстановления гиперповерхности

Описанная ранее стандартная процедура интерполяции может не подходить для обучения нейронных сетей RBF в некоторых классах задач из-за плохой обобщающей способности, вызванной следующей причиной: если количество точек данных обучающего множества значительно превышает количество степеней свободы самого физического процесса, а нам требуется иметь столько же радиальных базисных функций, сколько точек в обучающем множестве, задача оказывается *избыточно определенной* (overdetermined). Следовательно, сеть может завершить свою настройку в неверном положении, приняв во внимание сторонние шумы во входных данных, что, в свою очередь, станет причиной плохого обобщения [160].

Чтобы глубже понять проблему *избыточного подбора* (overfitting) и выбрать способы ее устранения, вспомним, что конструирование нейронной сети, призванной генерировать выходной сигнал в ответ на входной пример, эквивалентно обучению сети построению гиперповерхности (т.е. многомерному отображению), определяющей выходной вектор в терминах входного. Другими словами, *обучение рассматривается как задача реконструкции гиперповерхности на основе множества точек, которое может быть довольно разреженным.*

Согласно [554], [561] две задачи считаются *обратными* (inverse) друг другу, если формулировка каждой из них требует полного или частичного знания о другой. Обычно оказывается, что одна из задач уже решалась ранее и, возможно, даже более детально, чем другая. В таком случае первая задача называется *прямой* (direct problem), а вторая — *обратной* (inverse problem). Однако с точки зрения математики существует еще одно более важное отличие между прямой и обратной задачами. Задача может быть *плохо* или *хорошо обусловлена* (well-posed, ill-posed). Термин "хорошая обусловленность" используется в прикладной математике еще со времен Адамара (Hadamard) — с начала 1900-х годов. Для того чтобы объяснить эту терминологию, предположим, что область X и диапазон Y являются метрическими пространствами, которые связаны некоторым фиксированным, но неизвестным отображением f. Задача реконструкции отображения f считается хорошо обусловленной, если выполняются следующие три условия [561], [756], [1056].

1. *Существование* (existence). Для любого входного вектора $\mathbf{x} \in \mathbf{X}$ существует выходное значение $y = f(\mathbf{x})$, где $y \in \mathbf{Y}$.
2. *Уникальность* (uniqueness). Для любой пары входных векторов $\mathbf{x}, \mathbf{t} \in \mathbf{X}$ равенство $f(\mathbf{x}) = f(\mathbf{t})$ выполняется тогда и только тогда, когда $\mathbf{x} = \mathbf{t}$.
3. *Непрерывность* (continuity). Отображение считается непрерывным, если для любого $\varepsilon > 0$ существует $\delta = \delta(\varepsilon)$, такое, что из условия $\rho_x(\mathbf{x}, \mathbf{t}) < \delta$ вытекает, что $\rho_y(f(\mathbf{x}), f(\mathbf{y})) < \varepsilon$, где $\rho(\cdot, \cdot)$ — расстояние между двумя аргументами в соответствующих пространствах (рис. 5.3). Свойство непрерывности еще называют *устойчивостью* (stability).

Если какое-либо из этих условий не выполнено, задача считается *плохо обусловленной*. По существу, плохая обусловленность задачи означает, что даже большой набор данных может нести в себе удивительно малый объем информации о решении задачи.

В контексте рассматриваемой проблемы моделирование физических процессов, обеспечивающих генерирование обучающих данных (например, звуковой сигнал, изображение, эхо радара и т.п.), является хорошо обусловленной прямой задачей. Однако обучение на примере таких физических данных, рассматриваемое как задача восстановления гиперповерхности, является плохо обусловленной обратной задачей. Это объясняется следующими причинами. Во-первых, может быть нарушен критерий существования, так как не для каждого входного сигнала может существовать выходной. Во-вторых, информации, содержащейся в примерах, может быть недостаточно для корректной уникальной реконструкции отображения "вход-выход". Это значит, что критерий уникальности также может быть нарушен. В-третьих, неизбежное наличие шумов в данных обучения вносит неопределенность в восстанавливаемое отображение. В частности, если уровень шума во входном сигнале слишком высок, нейронная сеть в ответ на входной сигнал $\mathbf{x}$ из области $\mathbf{X}$ может давать на выходе сигнал, выходящий за пределы диапазона $\mathbf{Y}$. Другими словами, здесь может нарушаться критерий непрерывности. Если задача обучения не удовлетворяет критерию непрерывности, то вычисленное отображение входа на выход будет иметь мало общего с реальным решением задачи. Эту проблему никак нельзя обойти, если неизвестна какая-либо априорная информация об отображении. В этом контексте будет уместным вспомнить утверждение, сделанное в [609], относительно линейных дифференциальных операторов: "Недостаток информации нельзя восполнить никакой математической хитростью".

В следующем разделе рассматривается важный вопрос: как плохо обусловленную задачу сделать хорошо обусловленной с помощью методов регуляризации[3].

---

[3] Еще один подход к регуляризации на основе учета априорной информации в отображении — *байесовская интерполяция* (Bayesian interpolation). Подробное описание этого подхода содержится в [695], [696], [776].

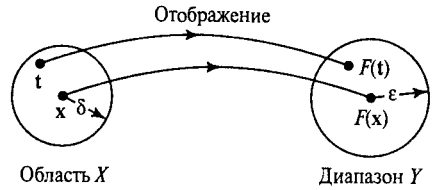


Рис. 5.3. Отображение (входной) области **X** в (выходной) диапазон **Y**

5.5. Теория регуляризации

В 1963 году Тихонов предложил новый метод, получивший название *регуляризации* (regularization) и предназначенный для решения плохо обусловленных задач[4]. В контексте задачи восстановления гиперповерхности главная идея регуляризации заключается в *стабилизации* решения с помощью некоторой вспомогательной неотрицательной функции, которая несет в себе априорную информацию о решении. Наиболее общей формой априорной информации является предположение о гладкости функции искомого отображения (т.е. решения задачи восстановления) в том смысле, что одинаковый входной сигнал соответствует одинаковому выходному.

Для примера возьмем множество пар данных "вход-выход" (т.е. пример обучения), доступных для аппроксимации и описываемых следующим образом.

$$\begin{aligned} &\text{Входной сигнал} : \mathbf{x}_i \in \Re^{m_0}, i = 1, 2, \ldots, N. \\ &\text{Желаемый отклик} : d_i \in \Re^1, i = 1, 2, \ldots, N. \end{aligned} \tag{5.20}$$

Обратите внимание, что предполагается одномерность выходного сигнала. Это допущение никак не ограничивает применимость описываемой здесь теории регуляризации. Обозначим функцию аппроксимации как $F(\mathbf{x})$, где (для упрощения выкладок) в списке аргументов опущен вектор весов $\mathbf{w}$. Теория регуляризации Тихонова в своем изначальном виде использует два слагаемых.

1. *Слагаемое стандартной ошибки* (standard error term). Первое слагаемое, обозначаемое $\mathbf{E}_s(F)$, описывает стандартную ошибку (расстояние между желаемым откликом d_i и фактическим выходным сигналом y_i для примера обучения $i = 1$,

[4] Открытие теории регуляризации обычно приписывается Тихонову [1055]. Однако аналогичный подход был предложен в 1962 году Филлипсом [836]. По этой причине в первоисточниках можно встретить термин *регуляризация Тихонова–Филлипса*.

Одна из форм регуляризации была описана в работе, где процесс сглаживания назывался *настройкой* (adjustment) наблюдений [1136].

Теория регуляризации подробно описывается в [561], [756], [1056].

2,..., N). В частности, можно определить

$$E_s(F) = \frac{1}{2}\sum_{i=1}^{N}(d_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]^2, \tag{5.21}$$

где множитель 1/2 введен из соображений совместимости с материалом предыдущих гла

2. *Слагаемое регуляризации* (regularization term). Это второе слагаемое, обозначаемое $\mathbf{E}_c(F)$, зависит от "геометрических" свойств функции аппроксимации $F(\mathbf{x})$. В частности, можно записать:

$$E_c(F) = \frac{1}{2}\|\mathbf{D}F\|^2, \tag{5.22}$$

где $\mathbf{D}$ — *линейный дифференциальный оператор* (linear differential operator). Априорная информация о форме решения (т.е. о функции отображения $F(\mathbf{x})$), включенная в дифференциальный оператор $\mathbf{D}$, обеспечивает его зависимость от конкретной задачи. Оператор $\mathbf{D}$ иногда еще называют *стабилизатором* (stabilizer), так как в задаче регуляризации он стабилизирует решение, делая его гладким и, таким образом, удовлетворяющим свойству непрерывности. Заметим, что гладкость предполагает непрерывность, но не наоборот.

Аналитический подход, используемый для работы с соотношением (5.22), основан на концепции *функционального пространства*[5] (function space), которая тесно связана с понятием *нормированного пространства*[6] (normed space) функций. В таком многомерном (строго говоря, бесконечномерном) пространстве непрерывная функция представляется *вектором*. Используя это геометрическое представление, можно увидеть интуитивную связь между матрицей и оператором линейного дифференцирования. Таким образом, анализ линейных систем можно свести к анализу линейных дифференциальных уравнений [609].

---

[5] Концепция функциональиого пространства впоследствии была развита Гильбертом (Hilbert) в его исследовании одного из классов интегральных уравнений. В то время как Фредгольм (Fredholm), основатель интеграла Фредгольма, сформулировал задачу на языке алгебры, Гильберт увидел ее тесную связь с задачами аналитической геометрии (поверхности второго порядка в многомерных Евклидовых пространствах) [609].

[6] Нормированным пространством называется пространство линейных векторов, в котором определена вещественная функция $||\mathbf{x}||$, называемая *нормой* $\mathbf{x}$. Норма $||\mathbf{x}||$ обладает следующими свойствами:

$$\begin{aligned} ||\mathbf{x}|| &> 0 \text{ , для } \mathbf{x} \neq 0, \\ ||\mathbf{0}|| &= 0 \text{ ,} \\ ||a\mathbf{x}|| &= |a| \cdot ||\mathbf{x}||, \text{ где } a \text{ — константа,} \\ ||\mathbf{x}+\mathbf{y}|| &\leq ||\mathbf{x}|| + ||\mathbf{y}||. \end{aligned}$$

Норма $||\mathbf{x}||$ играет роль "длины" вектора $\mathbf{x}$.

Символ $||\cdot||$ в выражении (5.22) обозначает норму в функциональном пространстве, к которому принадлежит $\mathbf{D}F(\mathbf{x})$. При обычных условиях используемое здесь функциональное пространство является *пространством* L_2, состоящим из всех действительных функций $f(\mathbf{x})$, $\mathbf{x} \in \Re^{m_0}$, для которых норма $||f(\mathbf{x})||^2$ является интегрируемой по Лебегу. Используемая здесь функция $f(\mathbf{x})$ обозначает фактическую функцию, описывающую моделируемый физический процесс, отвечающий за генерацию пар примеров обучения $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$[7].

Величиной, которую требуется минимизировать в теории регуляризации, является

$$E(F) = E_s(F) + \lambda E_c(F) = \frac{1}{2}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]^2 + \frac{1}{2}\lambda\|\mathbf{D}F\|^2, \tag{5.23}$$

где λ — положительное действительное число, называемое *параметром регуляризации* (regularization parameter); $\mathbf{E}(F)$ — *функционал Тихонова*. Функционал отображает функции (определенные в соответствующем функциональном пространстве) на ось действительных чисел. Аргминимум функционала Тихонова $\mathbf{E}(F)$ (т.е. решение задачи регуляризации) обозначается $F_\lambda(\mathbf{x})$.

В некотором смысле параметр регуляризации λ можно рассматривать как индикатор достаточности данного набора данных для определения решения$F_\lambda(\mathbf{x})$. В частности, крайний случай, $\lambda \to 0$, означает, что задача является безусловной и имеет решение $F_\lambda(\mathbf{x})$, целиком зависящее от примеров. Другой крайний случай, $\lambda \to \infty$, предполагает, что самого априорного ограничения на гладкость, представленного дифференциальным оператором $\mathbf{D}$, достаточно для определения решения $F_\lambda(\mathbf{x})$. Это может указывать также на недостоверное количество примеров. В практических приложениях параметр регуляризации λ принимает некоторое среднее значение между этими двумя крайними случаями. Этим определяется влияние на решение $F_\lambda(\mathbf{x})$ как априорной информации, так и данных обучающей выборки. Таким образом, слагаемое регуляризации $\mathbf{E}_c(F)$ представляет собой *функцию штрафа за сложность модели* (model complexity-penalty function), влияние которой на окончательное решение определяется параметром регуляризации λ.

---

[7] Строго говоря, требуется, чтобы функция $f(\mathbf{x})$, обеспечивающая генерирование данных, была членом *воспроизводящего ядра* (reproducing kernel) *Гильбертова пространства*, представленного в форме дельта-распределения Дирака δ [1041]. При этом требуется убывание и бесконечная непрерывная дифференцируемость дельта-функций этого распределения. Этому условию удовлетворяет классическое пространство тестовых функций $\mathbf{G}$ для теорин распределения Шварца (Schwarz theory of distributions) с коиечной $\mathbf{D}$-обусловленной нормой: $H_p = \{f \in \mathbf{G} : \|\mathbf{D}f\| < \infty\}$. Обычно, когда речь идет о Гильбертовом пространстве, вспоминают только о пространстве L_2, возможно, нз-за того, что последнее изоморфно любому Гильбертову пространству. Однако самым важным признаком Гильбертова пространства является норма, а изометрия (т.е. изоморфизм, сохраняющий норму) играет более важную роль, чем аддитивный изоморфизм [532]. Теория воспроизводящего ядра Гильбертова пространства показала, что кроме L_2 существует масса различных н вполне пригодных для практического использования Гильбертовых пространств. Подробно эта теория описывается в [533].

Считается, что теория регуляризации обеспечивает практическое решение дилеммы смещения и дисперсии, которая рассматривалась в главе 2. В частности, оптимальный выбор параметра регуляризации λ позволяет обеспечить удовлетворительное соотношение между *смещением и дисперсией модели* (model bias & model variance) при решении задачи обучения с учетом некоторой априорной информации.

Дифференциал Фреше функционала Тихонова

Теперь *принцип регуляризации* (regularization principle) можно записать в следующем виде.

Найти функцию $F_\lambda(\mathbf{x})$, *минимизирующую функционал Тихонова* $\mathbf{E}(F)$:

$$\mathbf{E}(F) = \mathbf{E}_s(F) + \lambda \mathbf{E}_c(F),$$

где $\mathbf{E}_s(F)$ *— слагаемое стандартной ошибки;* $\mathbf{E}_c(F)$ *— слагаемое регуляризации;* λ *— параметр регуляризации.*

Для того чтобы продолжить рассмотрение задачи минимизации функционала стоимости $\mathbf{E}(F)$, нужно задать правило оценки его дифференциала. Это можно осуществить с помощью *дифференциала Фреше* (Frechet differential). В элементарной математике касательной к кривой называется прямая, которая дает наилучшую аппроксимацию кривой в окрестности точки касания. Аналогично, дифференциал Фреше функционала можно интерпретировать как наилучшую локальную линейную аппроксимацию. Таким образом, дифференциал Фреше функционала $\mathbf{E}(F)$ формально определяется выражением [246], [250], [263]

$$dE(F,h) = \left[\frac{d}{d\beta}E(F+\beta h)\right]_{\beta=0}, \tag{5.24}$$

где $h(\mathbf{x})$ — фиксированная функция вектора $\mathbf{x}$. В выражении (5.24) используются обычные правила дифференцирования. Необходимым условием того, чтобы функция $F(\mathbf{x})$ была относительным экстремумом функционала $\mathbf{E}(F)$, является равенство нулю дифференциала Фреше $d\mathbf{E}(F,h)$ по F($\mathbf{x}$) для всех $h \in \mathbf{H}$, т.е.

$$d\mathbf{E}(F,h) = d\mathbf{E}_s(F,h) + \lambda d\mathbf{E}_c(F,h) = 0, \tag{5.25}$$

где $d\mathbf{E}_s(F,h)$ и $d\mathbf{E}_c(F,h)$ — дифференциалы Фреше функционалов $\mathbf{E}_s(F,h)$ и $\mathbf{E}_c(F,h)$ соответственно.

Оценивая дифференциал Фреше слагаемого стандартной ошибки $\mathbf{E}_s(F,h)$, определяемого формулой (5.21), имеем:

$$\begin{aligned} d\mathbf{E}_s(F,h) &= \left[\frac{d}{d\beta}\mathbf{E}_s(F+\beta h)\right]_{\beta=0} = \\ &= \left[\frac{1}{2}\frac{d}{d\beta}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i) - \beta h(\mathbf{x}_i)]^2\right]_{\beta=0} = \\ &= -\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i) - \beta h(\mathbf{x}_i)]\, h(\mathbf{x}_i)|_{\beta=0} = \\ &= -\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]h(\mathbf{x}_i). \end{aligned} \tag{5.26}$$

Здесь будет уместным привести *теорему Ритца о представлении* (Riesz representation theorem), которая формулируется следующим образом [246, 561].

Пусть f — ограниченный линейный функционал в Гильбертовом пространстве (т.е. в полном пространстве скалярных произведений)[8], *которое обозначается символом* $\mathbf{H}$. *Тогда существует единственное значение* $h_0 \in \mathbf{H}$, *такое, что*

$$f = (h, h_0)_{\mathbf{H}} \text{ для всех } h_0 \in \mathbf{H},$$

при этом

$$\|f\|_{\tilde{\mathbf{H}}} = \|h_0\|_{\mathbf{H}},$$

где $\tilde{\mathbf{H}}$ — *пространство, сопряженное Гильбертову пространству* $\mathbf{H}$.

---

[8] *Пространством скалярных произведений* (inner product space) называется линейное пространство векторов, в котором скалярное произведение двух векторов $\mathbf{u}$ и $\mathbf{v}$, обозначаемое как $(\mathbf{u}, \mathbf{v})$, обладает следующими свойствами:

$$\begin{aligned} (\mathbf{u},\mathbf{v}) &= (\mathbf{v},\mathbf{u}), \\ (a\mathbf{u},\mathbf{v}) &= a(\mathbf{u},\mathbf{v}),\ a - \text{константа}, \\ (\mathbf{u}+\mathbf{v},\mathbf{w}) &= (\mathbf{u},\mathbf{w}) + (\mathbf{v},\mathbf{w}), \\ (\mathbf{u},\mathbf{u}) &> 0 \text{ для } \mathbf{u} \neq 0. \end{aligned}$$

Пространство скалярных произведений $\mathbf{H}$ является *полным* (complete) и называется Гильбертовым пространством, если любая последовательность Коши, выбранная из этого пространства, сходится по норме к какому-либо пределу. Последовательность векторов $\{\mathbf{x}_n\}$ называется *последовательностью Коши*, если для любого $\varepsilon > 0$ существует число N, такое, что [246]

$$\|\mathbf{x}_m - \mathbf{x}_n\| < \varepsilon \text{ для всех } (n,m) > M.$$

Символом $(\cdot,\cdot)_H$ здесь обозначается *скалярное произведение* двух функций в пространстве **H**. В свете теоремы Ритца о представлении дифференциал Фреше $d\mathbf{E}_s(F,h)$ из (5.26) можно переписать в эквивалентном виде:

$$dE_s(F,h) = -\left(h, \sum_{i=1}^{N}(d_i - F)\delta_{\mathbf{x}_i}\right)_{\mathbf{H}}, \tag{5.27}$$

где $\delta_{\mathbf{x}_i}$ — *дельта-распределение Дирака* (Dirac delta distribution) вектора $\mathbf{x}$, с центром в точке $\mathbf{x}_i$, т.е.

$$\delta_{\mathbf{x}_i} = \delta(\mathbf{x} - \mathbf{x}_i). \tag{5.28}$$

Аналогично рассмотрим следующую оценку дифференциала Фреше слагаемого регуляризации $\mathbf{E}_c(F)$ из (5.22):

$$\begin{aligned} dE_c(F,h) &= \frac{d}{d\beta}E_c(F+\beta h)\,|_{\beta=0} = \frac{1}{2}\frac{d}{d\beta}\int_{\Re^{m_0}}(\mathbf{D}[F+\beta h])^2 d\mathbf{x}\,|_{\beta=0} = \\ &= \int_{\Re^{m_0}}\mathbf{D}[F+\beta h]\mathbf{D}h d\mathbf{x}\,|_{\beta=0} = \int_{\Re^{m_0}}\mathbf{D}F\mathbf{D}h d\mathbf{x} = (\mathbf{D}F, \mathbf{D}h)_{\mathbf{H}}, \end{aligned} \tag{5.29}$$

где $(\mathbf{D}F,\ \mathbf{D}h)_{\mathbf{H}}$ — скалярное произведение функций $\mathbf{D}h(\mathbf{x})$ и $\mathbf{D}F(\mathbf{x})$, которые являются результатом применения оператора дифференцирования $\mathbf{D}$ к $h(\mathbf{x})$ и $F(\mathbf{x})$ соответственно.

Уравнение Эйлера–Лагранжа

Для данного оператора $\mathbf{D}$ можно найти такой уникально определенный *сопряженный оператор* $\tilde{\mathbf{D}}$, что для любой пары дифференцируемых функций $u(\mathbf{x})$ и $v(\mathbf{x})$, удовлетворяющих соответствующим граничным условиям, можно записать [609]:

$$\int_{\Re^m} u(\mathbf{x})\mathbf{D}v(\mathbf{x})d\mathbf{x} = \int_{\Re^m} v(\mathbf{x})\tilde{\mathbf{D}}u(\mathbf{x})d\mathbf{x}. \tag{5.30}$$

Равенство (5.30) получило название *тождества Грина* (Green's identity). Оно подводит математический базис под определение сопряженного оператора $\tilde{\mathbf{D}}$ в терминах данного дифференциала $\mathbf{D}$. Если рассматривать $\mathbf{D}$ как матрицу, сопряженный оператор $\tilde{\mathbf{D}}$ играет роль, аналогичную транспонированной матрице.

Сравнивая тождество (5.30) с предпоследней выкладкой (5.29), можно установить следующие соответствия:

$$\begin{aligned} u(\mathbf{x}) &= \mathbf{D}F(\mathbf{x}), \\ \mathbf{D}v(\mathbf{x}) &= \mathbf{D}h(\mathbf{x}). \end{aligned}$$

Используя тождество Грина, (5.29) можно переписать в эквивалентной форме:

$$d\mathbf{E}_c(F,h) = \int_{\Re^{m_0}} h(\mathbf{x})\tilde{\mathbf{D}}\mathbf{D}F(\mathbf{x})d\mathbf{x} = (h, \tilde{\mathbf{D}}\mathbf{D}F)_H, \tag{5.31}$$

где $\tilde{\mathbf{D}}$ — оператор, сопряженный $\mathbf{D}$.

Возвращаясь к условию экстремума, описанному выражением (5.25), и подставляя в него дифференциалы Фреше из (5.27) и (5.31), дифференциал Фреше $d\mathbf{E}(F,h)$ можно представить в следующем виде:

$$d\mathbf{E}(F,h) = \left(h, \left[\tilde{\mathbf{D}}\mathbf{D}F - \frac{1}{\lambda}\sum_{i=1}^{N}(d_i - F)\delta_{\mathbf{x}_i}\right]\right)_{\mathbf{H}}. \tag{5.32}$$

Так как параметру регуляризации λ обычно присваивается некоторое значение из открытого интервала $(0, \infty)$, дифференциал Фреше равен нулю для любого $h(\mathbf{x})$ в пространстве $\mathbf{H}$ тогда и только тогда, когда выполняется следующее условие в смысле распределения:

$$\tilde{\mathbf{D}}\mathbf{D}F_\lambda - \frac{1}{\lambda}\sum_{i=1}^{N}(d_i - F)\delta_{\mathbf{x}_i} = 0,$$

или, что эквивалентно,

$$\tilde{\mathbf{D}}\mathbf{D}F_\lambda(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]\delta(\mathbf{x} - \mathbf{x}_i). \tag{5.33}$$

Уравнение (5.33) называется *уравнением Эйлера–Лагранжа* (Euler-Lagrange equation) для функционала Тихонова $\mathbf{E}(F)$. Оно является необходимым условием существования экстремума функционала Тихонова в точке $F_\lambda(\mathbf{x})$ [246].

Функция Грина

Соотношение (5.33) представляет собой уравнение в частных производных функции аппроксимации F. Решение этого уравнения состоит в интегральном преобразовании его правой части.

Пусть $G(\mathbf{x}, \boldsymbol{\xi})$ — функция, в которой векторы $\mathbf{x}$ и $\boldsymbol{\xi}$ используются для разных целей: вектор $\mathbf{x}$ используется как параметр, а $\boldsymbol{\xi}$ — как аргумент. Предполагается, что для данного оператора линейного дифференцирования $\mathbf{L}$ функция $G(\mathbf{x}, \boldsymbol{\xi})$ удовлетворяет следующему условию [217].

1. Для фиксированного $\boldsymbol{\xi}$ $G(\mathbf{x}, \boldsymbol{\xi})$ является функцией от $\mathbf{x}$ и удовлетворяет граничным условиям.

2. Во всех точках, за исключением $\mathbf{x} = \boldsymbol{\xi}$, все производные $G(\mathbf{x}, \boldsymbol{\xi})$ по $\mathbf{x}$ являются непрерывными. Количество производных определяется порядком оператора $\mathbf{L}$.
3. Если рассматривать $G(\mathbf{x}, \boldsymbol{\xi})$ как функцию от $\mathbf{x}$, она удовлетворяет уравнению в частных производных

$$\mathbf{L}G(\mathbf{x}, \boldsymbol{\xi}) = 0, \tag{5.34}$$

всюду, за исключением точки $\mathbf{x} = \boldsymbol{\xi}$, где она имеет особенность. Это значит, что функция $G(\mathbf{x}, \boldsymbol{\xi})$ удовлетворяет уравнению в частных производных (в смысле распределения)

$$\mathbf{L}G(\mathbf{x}, \boldsymbol{\xi}) = \delta(\mathbf{x} - \boldsymbol{\xi}), \tag{5.35}$$

где, как говорилось ранее, $\delta(\mathbf{x} - \boldsymbol{\xi})$ — дельта-функция Дирака с центром в точке $\mathbf{x} = \boldsymbol{\xi}$.

Описанная таким образом функция $G(\mathbf{x}, \boldsymbol{\xi})$ называется *функцией Грина* (Green's function) для оператора дифференцирования $\mathbf{L}$. Функция Грина играет ту же роль для оператора дифференцирования, которую в матричном исчислении играет обратная матрица.

Пусть $\varphi(\mathbf{x})$ — непрерывная или кусочно-непрерывная функция аргумента $\mathbf{x} \in \Re^{m_0}$. Тогда функция

$$F(\mathbf{x}) = \int_{\Re^{m_0}} G(\mathbf{x}, \boldsymbol{\xi})\varphi(\boldsymbol{\xi})d\boldsymbol{\xi} \tag{5.36}$$

является решением дифференциального уравнения

$$\mathbf{L}F(\mathbf{x}) = \varphi(\mathbf{x}), \tag{5.37}$$

где $G(\mathbf{x},\boldsymbol{\xi})$ — функция Грина для линейного оператора дифференцирования $\mathbf{L}$ [217].

Для того чтобы доказать правильность выбора $F(\mathbf{x})$ в качестве решения уравнения (5.37), применим к выражению (5.36) оператор дифференцирования $\mathbf{L}$, в результате чего получим:

$$\mathbf{L}F(\mathbf{x}) = \mathbf{L}\int_{\Re^{m_0}} G(\mathbf{x}, \boldsymbol{\xi})\varphi(\boldsymbol{\xi})d\boldsymbol{\xi} = \int_{\Re^{m_0}} \mathbf{L}G(\mathbf{x}, \boldsymbol{\xi})\varphi(\boldsymbol{\xi})d\boldsymbol{\xi}. \tag{5.38}$$

Оператор дифференцирования $\mathbf{L}$ использует $\boldsymbol{\xi}$ как константу, работая с функцией Грина только как с функцией от аргумента $\mathbf{x}$. Используя в выражении (5.38) формулу (5.35), получим:

$$\mathbf{L}F(x) = \int_{\Re^{m_0}} \delta(\mathbf{x} - \boldsymbol{\xi})\varphi(\boldsymbol{\xi})d\boldsymbol{\xi}.$$

И, в заключение, используя просеивающее свойство дельта-функции Дирака, а именно

$$\int_{\Re^{m_0}} \varphi(\boldsymbol{\xi})\delta(\mathbf{x}-\boldsymbol{\xi})d\boldsymbol{\xi} = \varphi(\mathbf{x}),$$

получим $\mathbf{L}F(\mathbf{x}) = \varphi(\mathbf{x})$, что и было описано в выражении (5.37).

Решение задачи регуляризации

Возвращаясь к нашей задаче, а именно к решению уравнения Эйлера–Лагранжа (5.33), определим:

$$\mathbf{L} = \tilde{\mathbf{D}}\mathbf{D} \tag{5.39}$$

и

$$\varphi(\boldsymbol{\xi}) = \frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]\delta(\boldsymbol{\xi}-\mathbf{x}_i). \tag{5.40}$$

Тогда можно использовать (5.36) и записать:

$$F_\lambda(\mathbf{x}) = \int_{\Re^{m_0}} G(\mathbf{x},\boldsymbol{\xi})\left\{\frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]\delta(\boldsymbol{\xi}-\mathbf{x}_i)\right\}d\boldsymbol{\xi} =$$
$$= \frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]\int_{\Re^{m_0}} G(\mathbf{x},\boldsymbol{\xi})\delta(\boldsymbol{\xi}-\mathbf{x}_i)d\boldsymbol{\xi},$$

где в последней строке изменен порядок интегрирования и суммирования. Используя свойство просеивания дельта-функции Дирака, получим требуемое решение уравнения Эйлера–Лагранжа:

$$F_\lambda(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]G(\mathbf{x},\mathbf{x}_i). \tag{5.41}$$

Выражение (5.41) означает, что минимизирующее решение $F_\lambda(\mathbf{x})$ задачи регуляризации является линейной суперпозицией N функций Грина. Векторы $\mathbf{x}_i$ представляют собой *центры разложения* (center of expansion), а веса $[d_i - F(\mathbf{x}_i)]/\lambda$ — *коэффициенты разложения* (coefficient of expansion). Другими словами, решение задачи регуляризации лежит в N-мерном подпространстве пространства гладких функций, а множество функций Грина $\{G(\mathbf{x}, \mathbf{x}_i)\}$, с центром в $\mathbf{x}_i, i = 1, 2, \ldots, N$, определяет базис этого подпространства [847]. Обратите внимание, что коэффициенты разложе-

ния в (5.41) являются, во-первых, линейными относительно ошибки оценки, определенной как разность между желаемым откликом d_i и соответствующим выходным сигналом $F(\mathbf{x}_i)$, вычисляемым сетью, и, во-вторых, обратно пропорциональными параметру регуляризации λ.

Определение коэффициентов разложения

Следующим вопросом является определение коэффициентов разложения в выражении (5.41). Пусть

$$w_i = \frac{1}{\lambda}[d_i - F(\mathbf{x}_i)], \quad i = 1, 2, \ldots, N. \tag{5.42}$$

Тогда выражение для минимизирующего решения (5.41) можно упростить:

$$F_\lambda(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{N} w_i G(\mathbf{x}, \mathbf{x}_i). \tag{5.43}$$

Вычисляя (5.43) в точке $\mathbf{x}_j, j = 1, 2, \ldots, N$, получим:

$$F_\lambda(\mathbf{x}_j) = \sum_{i=1}^{N} w_i G(\mathbf{x}_j, \mathbf{x}_i), \quad j = 1, 2, \ldots, N. \tag{5.44}$$

Введем следующие определения:

$$\mathbf{F}_\lambda = [F_\lambda(\mathbf{x}_1), F_\lambda(\mathbf{x}_2), \ldots, F_\lambda(\mathbf{x}_N)]^T, \tag{5.45}$$

$$\mathbf{d} = [d_1, d_2, \ldots, d_N]^T, \tag{5.46}$$

$$\mathbf{G} = \begin{bmatrix} G(\mathbf{x}_1,\mathbf{x}_1) & G(\mathbf{x}_1,\mathbf{x}_2) & \ldots & G(\mathbf{x}_1,\mathbf{x}_N) \\ G(\mathbf{x}_2,\mathbf{x}_1) & G(\mathbf{x}_2,\mathbf{x}_2) & \ldots & G(\mathbf{x}_2,\mathbf{x}_N) \\ \ldots & \ldots & \ldots\ldots & \\ G(\mathbf{x}_N,\mathbf{x}_1) & G(\mathbf{x}_N,\mathbf{x}_2) & \ldots & G(\mathbf{x}_N,\mathbf{x}_{1N}) \end{bmatrix}, \tag{5.47}$$

$$\mathbf{w} = [w_1, w_2, \ldots, w_N]^T. \tag{5.48}$$

Теперь (5.42) и (5.44) можно переписать в матричной форме:

$$\mathbf{w} = \frac{1}{\lambda}(\mathbf{d} - \mathbf{F}_\lambda) \tag{5.49}$$

и

$$\mathbf{F}_\lambda = \mathbf{G}\mathbf{w}. \tag{5.50}$$

Подставляя выражение для $\mathbf{F}_\lambda$ из (5.50) и (5.49) и переставляя слагаемые, получим:

$$(\mathbf{G} + \lambda\mathbf{I})\mathbf{w} = \mathbf{d}, \tag{5.51}$$

где $\mathbf{I}$ — единичная матрица размерности $N \times N$. Матрица $\mathbf{G}$ называется *матрицей Грина* (Green's matrix).

Оператор дифференцирования $\mathbf{L}$, определяемый выражением (5.39), является *самосопряженным* (self-adjoint). Это значит, что оператор, сопряженный $\mathbf{L}$, равен ему самому. Отсюда следует, что ассоциированная функция Грина $\mathbf{G}(\mathbf{x}, \mathbf{x}_i)$ является *симметричной*, т.е.

$$G(\mathbf{x}_i, \mathbf{x}_j) = G(\mathbf{x}_j, \mathbf{x}_i) \text{ для всех } i \text{ и } j. \tag{5.52}$$

Уравнение (5.52) означает, что позиции двух точек $\mathbf{x}$ и $\boldsymbol{\xi}$ можно поменять местами, при этом значение функции Грина $\mathbf{G}(\mathbf{x}, \boldsymbol{\xi})$ не изменится. Аналогично, матрица Грина, определяемая выражением (5.52), является *симметричной*, т.е.

$$\mathbf{G}^T = \mathbf{G}. \tag{5.53}$$

Теперь можно применить теорему об интерполяции, сформулированную в разделе 5.3, в контексте матрицы интерполяции $\boldsymbol{\Phi}$. Во-первых, заметим, что матрица Грина $\mathbf{G}$ играет в теории регуляризации роль, аналогичную той, которую матрица $\boldsymbol{\Phi}$ играет в теории интерполяции на основе сетей RBF. Обе матрицы, $\boldsymbol{\Phi}$ и $\mathbf{G}$, являются симметричными и имеют размерность $N \times N$. Следовательно, можно утверждать, что матрица $\mathbf{G}$ для некоторых классов функций Грина является положительно определенной при условии, что точки $\mathbf{x}_1, \ldots, \mathbf{x}_N$ различны. Классы функций Грина, охватываемые теоремой Мичелли, включают обратные параболические функции и функции Гаусса, но не включают параболические функции. На практике всегда можно выбрать значение λ, достаточно большое для обеспечения положительной определенности матрицы $(\mathbf{G} + \lambda\mathbf{I})$ и, таким образом, для возможности ее инвертирования. А это, в свою очередь, значит, что система линейных уравнений (5.51) имеет единственное решение, определяемое следующим образом [847]:

$$\mathbf{w} = (\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{d}. \tag{5.54}$$

Таким образом, выбрав оператор дифференцирования $\mathbf{D}$ и имея набор функций Грина $G(\mathbf{x}_i, \mathbf{x}_j)$, где $i = 1, 2, \ldots, N$, соотношение (5.44) можно использовать для получения вектора весов $\mathbf{w}$, соответствующего вектору желаемого отклика $\mathbf{d}$ и данному значению параметра регуляризации λ.

В заключение можно утверждать, что решение задачи регуляризации задается следующим разложением[9]:

$$F_{\lambda}(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\mathbf{x}, \mathbf{x}_i), \tag{5.55}$$

где $G(\mathbf{x}, \mathbf{x}_i)$ — функция Грина для самосопряженного оператора $\mathbf{L} = \tilde{\mathbf{D}}\mathbf{D}$; w_i — i-й элемент вектора весов $\mathbf{w}$. Эти два равенства задаются выражениями (5.35) и (5.54) соответственно. Уравнение (5.55) означает следующее [847].

- Регуляризационный подход эквивалентен разложению решений в терминах множества функций Грина, характеристики которых зависят от принятой формы оператора дифференцирования $\mathbf{D}$ и соответствующих граничных условий.
- Количество функций Грина, используемых в разложении, равно количеству примеров, используемых при обучении.

Однако нельзя не заметить, что решение задачи регуляризации, представленное выражением (5.55), является неполным, так как представляет собой лишь решение по модулю $g(\mathbf{x})$, лежащее в нуль-пространстве оператора $\mathbf{D}$ [847]. Это объясняется тем, что все функции, лежащие в нуль-пространстве оператора $\mathbf{D}$, являются "невидимыми" для слагаемого сглаживания $||\mathbf{D}F||^2$ в функционале стоимости $\mathbf{E}(F)$ (5.23). Под нуль-пространством оператора $\mathbf{D}$ понимается множество *всех* функций $g(\mathbf{x})$, для которых выполняется $\mathbf{D}g = 0$. Точная форма дополнительного слагаемого $g(\mathbf{x})$ определяется для конкретной задачи в том смысле, что она зависит от выбранного стабилизатора и граничных условий данной задачи. Например, оно не требуется для стабилизатора $\mathbf{D}$, соответствующего колоколообразной функции Грина, такой как функция Гаусса. Поскольку включение дополнительного слагаемого не влияет на основные выводы, в дальнейшем мы его учитывать не будем.

---

[9] В [361] представлен другой метод вывода формулы (5.55) на основе связывания слагаемого регуляризации $\mathbf{E}_C(F)$ с гладкостью функции аппроксимации $\mathrm{F}(\mathbf{x})$ непосредственно.

Гладкость в этой работе рассматривалась как мера колебания функции. В частности, считалось, что одна функция является более гладкой, нежели другая, если ее колебания меньше по амплитуде. Другими словами, мерой гладкости функции считалась частота ее колебаний. Предполагая именно такую меру гладкости, обозначим термином $F(\mathbf{s})$ многомерное преобразование Фурье функции $F(\mathbf{x})$, где $\mathbf{s}$ — многомерная переменная преобразования. Пусть $H(\mathbf{s})$ — некоторая положительная функция, которая стремится к нулю при стремлении нормы аргумента $\mathbf{s}$ к бесконечности, так что $1/H(\mathbf{s})$ представляет собой *фильтр верхних частот* (high-pass filter). Согласно вышеупомянутой работе можно определить гладкий функционал для слагаемого регуляризации в следующем виде:

$$\mathbf{E}_C(F) = \frac{1}{2}\int_{\Re^{m_0}} \frac{|F(\mathbf{s})|^2}{H(\mathbf{s})} d\mathbf{s},$$

где m_0 — размерность вектора $\mathbf{x}$. Согласно *теореме Парсеваля* (Parseval's theorem) из теории Фурье этот функционал является мерой мощности выходного сигнала фильтра верхних частот $1/H(\mathbf{s})$. Таким образом, переводя теорию регуляризации в плоскость теории Фурье и используя свойства преобразования Фурье, в вышеупомянутой работе была выведена формула (5.55).

Характеристика функции Грина $G(\mathbf{x}, \mathbf{x}_i)$ для заданного центра $\boldsymbol{x}_i$ зависит только от формы стабилизатора **D** (согласно априорному предположению касательно искомого отображения "вход-выход"). Если стабилизатор **D** является *инвариантным к преобразованиям* (translationally invariant), то функция Грина $G(\mathbf{x}, \mathbf{x}_i)$ с центром в $\mathbf{x}_i$, будет зависеть только от разности между аргументами $\mathbf{x}$ и $\mathbf{x}_i$, т.е.

$$G(\mathbf{x}, \mathbf{x}_i) = G(\mathbf{x} - \mathbf{x}_i). \tag{5.56}$$

Если же стабилизатор **D** *инвариантен* как *к преобразованиям*, так и *к поворотам* (rotationally), то функция Грина $G(\mathbf{x}, \mathbf{x}_i)$ будет зависеть только от Евклидовой нормы вектора разности ($\mathbf{x}$-$\mathbf{x}_i$), т.е.

$$G(\mathbf{x}, \mathbf{x}_i) = G(||\mathbf{x} - \mathbf{x}_i||). \tag{5.57}$$

При этих условиях функция Грина должна быть радиальной базисной функцией. В таком случае решение задачи регуляризации (5.55) принимает следующую частную форму [847]:

$$F_\lambda(\mathbf{x}) = \frac{1}{\lambda} \sum_{i=1}^{N} w_i G(\|\mathbf{x} - \mathbf{x}_i\|). \tag{5.58}$$

Решение (5.58) определяет пространство линейных функций, зависящее от известных точек данных и с учетом Евклидова расстояния.

Решение, описанное выражением (5.58), называется *строгой интерполяцией* (strict interpolation), так как для интерполяции функции $F(\mathbf{x})$ используются все N точек, доступных для обучения. Однако при этом важно отметить, что это решение в корне отличается от решения (5.11): оно регуляризировано с помощью определения (5.54) для вектора весов $\mathbf{w}$. Только при достижении параметром регуляризации значения, равного нулю, эти два решения становятся идентичными.

Многомерные функции Гаусса

Функция Грина $G(\mathbf{x}, \mathbf{x}_i)$, линейный дифференциальный оператор **D** которой инвариантен к трансформациям и вращению и удовлетворяет условию (5.57), представляет на практике большой интерес. Примером такой функции Грина является *многомерная функция Гаусса* (multivariate Gaussian function), определяемая следующим образом:

$$G(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{1}{2\delta_i^2} \|\mathbf{x} - \mathbf{x}_i\|^2\right), \tag{5.59}$$

где $\mathbf{x}_i$ — центр функции; σ_i — ее ширина. Самосопряженный оператор $\mathbf{L} = \tilde{\mathbf{D}}\mathbf{D}$ определяет функцию Грина в (5.59), имеющую вид [847]

$$\mathbf{L} = \sum_{n=0}^{\infty} (-1)^n \alpha_n \nabla^{2n}, \tag{5.60}$$

где

$$\alpha_n = \frac{\sigma_i^{2n}}{n! 2^n}, \tag{5.61}$$

а ∇^{2n} — итерированный оператор Лапласа для m_0 измерений

$$\nabla^2 = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \ldots + \frac{\partial^2}{\partial x_{m_0}^2}. \tag{5.62}$$

При стремлении количества слагаемых в (5.60) к бесконечности $\mathbf{L}$ становится дифференциальным оператором в обычном смысле. Поэтому оператор $\mathbf{L}$ в формуле (5.60) называется *псевдодифференциальным оператором* (pseudo-differential operator). Исходя из определения $\mathbf{L} = \tilde{\mathbf{D}}\mathbf{D}$, можно сделать вывод, что оператор $\mathbf{D}$ и сопряженный ему $\tilde{\mathbf{D}}$ можно представить в виде[10]

$$\begin{aligned} \mathbf{D} &= \sum_n \alpha_n^{1/2} \left(\frac{\partial}{\partial x_1} + \frac{\partial}{\partial x_2} + \ldots + \frac{\partial}{\partial x_{m_0}} \right)^n = \\ &= \sum_{a+b+\ldots+k=n} \alpha_n^{1/2} \frac{\partial^n}{\partial x_1^a \partial x_2^b \ldots \partial x_{m_0}^k}, \end{aligned} \tag{5.63}$$

$$\begin{aligned} \tilde{\mathbf{D}} &= \sum_n (-1)^n \alpha_n^{1/2} \left(\frac{\partial}{\partial x_1} + \frac{\partial}{\partial x_2} + \ldots + \frac{\partial}{\partial x_{m_0}} \right)^n = \\ &= \sum_{a+b+\ldots+k=n} (-1)^n \alpha_n^{1/2} \frac{\partial^n}{\partial x_1^a \partial x_2^b \ldots \partial x_{m_0}^k}. \end{aligned} \tag{5.64}$$

---

[10] Самой общей формой оператора дифференцирования является следующая:

$$\mathbf{D} = p(x_1, x_2, \ldots, x_{m_0}) \frac{\partial^n}{\partial x_1^a \partial x_2^b \ldots \partial x_{m_0}^k}, \quad a + b + \ldots + k = n,$$

где $x_1, x_2, \ldots, x_{m_0}$ — элементы вектора $\mathbf{x}$; $p(x_1, x_2, \ldots, x_{m_0})$ — некоторая функция этих элементов. Сопряженный оператор в этом случае выглядит следующим образом [757]:

$$\tilde{\mathbf{D}} = (-1)^n \frac{\partial^n}{\partial x_1^a \partial x_2^b \ldots \partial x_{m_0}^k} [p(x_1, x_2, \ldots, x_m)], \quad a + b + \ldots + k = n.$$

Таким образом, регуляризированное решение (5.58) достигается за счет использования стабилизатора, включающего все возможные частные производные.

Используя выражения (5.59)-(5.61) в (5.35) и приравнивая $\boldsymbol{\xi}$ к $\mathbf{x}_i$, можно записать:

$$\sum_n (-1)^n \frac{\sigma_i^{2n}}{n!2^n} \nabla^{2n} \exp\left(-\frac{1}{2\sigma_i^2} \|\mathbf{x} - \mathbf{x}_i\|^2\right) = \delta(\mathbf{x} - \mathbf{x}_i). \tag{5.65}$$

Определяя функцию Грина $G(\mathbf{x}, \mathbf{x}_i)$ в специальном виде (5.59), регуляризируемое решение (5.55) принимает форму линейной суперпозиции многомерных функций Гаусса:

$$F_\lambda(\mathbf{x}) = \sum_{i=1}^{N} w_i \exp\left(-\frac{1}{2\sigma_i^2} \|\mathbf{x} - \mathbf{x}_i\|^2\right), \tag{5.66}$$

где сами линейные веса w_i определяются по формуле (5.42).

В выражении (5.66) отдельные слагаемые функции Гаусса, определяющие функцию аппроксимации $F(\mathbf{x})$, в качестве аргументов содержат разные переменные. Для упрощения изложения в $F(\mathbf{x})$ часто принимается условие $\sigma_i = \sigma$ для всех i. Несмотря на то что определенные таким образом функции имеют несколько ограниченный вид, они остаются универсальными аппроксиматорами [815].

5.6. Сети регуляризации

Для реализации разложения регуляризованной функции аппроксимации $F_\lambda(\mathbf{x})$, представленной в (5.55) в терминах функции Грина $G(\mathbf{x}, \mathbf{x}_i)$ с центром в точке $\mathbf{x}_i$, можно использовать нейросетевую структуру, показанную на рис. 5.4. По очевидным причинам такие сети называются *сетями регуляризации* (regularization network) [847]. Как и сеть, представленная в разделе 5.1, эта сеть имеет три слоя. Входной слой состоит из входных узлов, количество которых равно размерности m_0 вектора входного сигнала $\mathbf{x}$ (т.е. количеству независимых переменных в задаче). Второй слой является скрытым. Он состоит из нелинейных элементов, которые *непосредственно* связаны со всеми узлами входного слоя. Для каждой точки данных $\mathbf{x}_i$ ($i = 1, 2, \ldots, N$, где N — размер множества примеров обучения) существует свой скрытый узел. Функциями активации отдельных узлов скрытого слоя являются функции Грина. Следовательно, выходной сигнал i-го нейрона скрытого слоя определяется как $G(\mathbf{x}, \mathbf{x}_i)$. Выходной слой состоит из единственного линейного нейрона, связанного со всеми узлами скрытого слоя. Под "линейностью" подразумевается то, что его выход является линейно-взвешенной суммой всех выходных сигналов скрытого слоя. Веса выходного слоя являются неизвестными коэффициентами разложения, определяемого в терминах функций Грина $G(\mathbf{x}, \mathbf{x}_i)$ и параметра регуляризации λ (см. (5.54)). На рис. 5.4

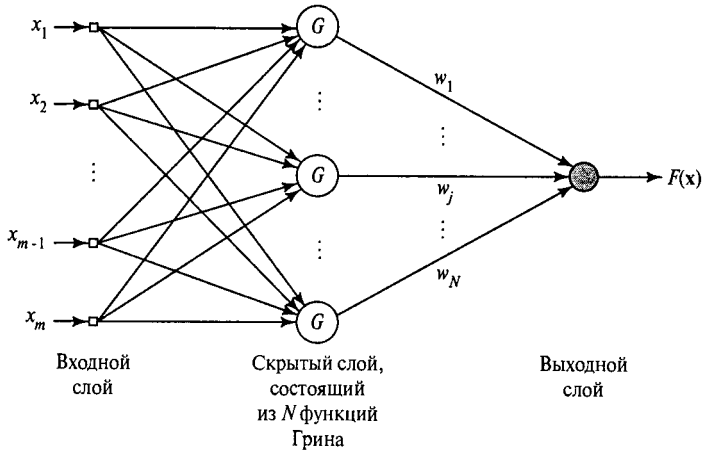


Рис. 5.4. Сеть регуляризации

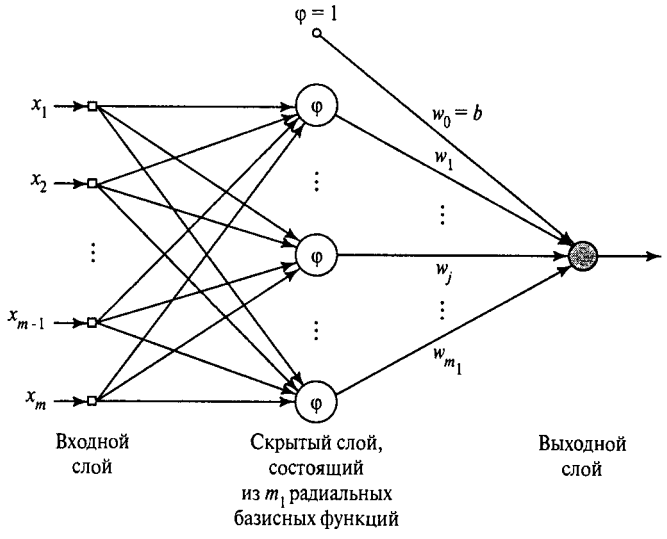


Рис. 5.5. Сеть на основе радиальных базисных функций

показана архитектура сети регуляризации с одним выходом. На рисунке видно, что такая архитектура может быть расширена для выходного сигнала любой размерности.

В сети регуляризации, показанной на рис. 5.5, предполагается, что функция Грина $G(\mathbf{x}, \mathbf{x}_i)$ является *положительно-определенной* для всех $\mathbf{x}_i$. При выполнении этого условия (например, если функция Грина имеет вид функции Гаусса (5.59)) решение, генерируемое сетью, будет являться оптимальной интерполяцией в смысле минимизации функционала $\mathbf{E}(F)$. Более того, с точки зрения теории аппроксимации сети регуляризации обладают следующими положительными свойствами [847].

1. Сеть регуляризации является универсальным аппроксиматором в том смысле, что при большом количестве скрытых элементов она способна довольно хорошо аппроксимировать любую непрерывную функцию на компактном подмножестве из $\Re^{m_0}$.
2. Так как схема аппроксимации, вытекающая из теории регуляризации, является линейной относительно неизвестных коэффициентов, то сети регуляризации обладают свойством *наилучшей аппроксимации* (best-approximation property). Это значит, что для неизвестной линейной функции f всегда существует такой набор коэффициентов, который аппроксимирует функцию f лучше любого другого набора.
3. Решение, обеспечиваемое сетью регуляризации, является *оптимальным*. Под оптимальностью здесь понимается то, что сеть регуляризации минимизирует функционал, измеряющий удаленность решения от своего истинного значения, представленного примерами обучения.

5.7. Обобщенные сети на основе радиальных базисных функций

Однозначное соответствие между данными обучения $\mathbf{x}_i$ и функциями Грина $G(\mathbf{x}, \mathbf{x}_i)$ для $i = 1, 2, \ldots, N$ явилось основой создания сети регуляризации, которая в вычислительном смысле при больших значениях N является очень ресурсоемкой. В частности, вычисление линейных весов сети (т.е. коэффициентов разложения в (5.55)) требует реализации операции обращения матрицы размерности $N \times N$, вычислительная сложность которой возрастает пропорционально N^3. Более того, для больших матриц ухудшается *обусловленность*, поскольку *число обусловленности* (condition number) матрицы определяется как отношение ее максимального собственного числа к минимальному. Для того чтобы обойти эти вычислительные трудности, необходимо уменьшить сложность сети, т.е. найти некоторую аппроксимацию регуляризированного решения.

Такой подход предполагает поиск субоптимального решения в пространстве меньшей размерности, аппроксимирующего регуляризированное решение (5.55). Для этого используется стандартный прием, получивший в вариационных задачах название *метода Галеркина* (Galerkin's method). Согласно этой технологии приближенное решение $F^*(\mathbf{x})$ находится как разложение по конечному базису, т.е. [847]

$$F^*(\mathbf{x}) = \sum_{i=1}^{m_1} w_i \varphi_i(\mathbf{x}), \tag{5.67}$$

где $\{\varphi_i(\mathbf{x}) | \quad i = 1, 2, \ldots, m_1\}$ — новое множество базисных функций, которые без потери общности предполагаются линейно-независимыми, а w_i — новое множество весов. Обычно количество базисных функций меньше числа точек данных обучения

(т.е. $m_1 \leq N$). Для радиальных базисных функций можно принять

$$\varphi_i(\mathbf{x}) = G(||\mathbf{x} - \mathbf{t}_i||), i = 1, 2, \ldots, m_1, \tag{5.68}$$

где множество центров $\{\mathbf{t}_i | \quad i = 1, 2, \ldots, m_1\}$ необходимо определить. Только такой выбор базисных функций гарантирует, что в случае $m_1 = N$ и

$$\mathbf{t}_i = \mathbf{x}_i, i = 1, 2, \ldots, N$$

корректное решение (5.58) будет полностью восстановлено. Таким образом, подставляя (5.68) в (5.67), можно переопределить $F^*(\mathbf{x})$ следующим образом:

$$F^*(\mathbf{x}) = \sum_{i=1}^{m_1} w_i G(\mathbf{x}, \mathbf{t}_i) = \sum_{i=1}^{m_1} w_i G(\|\mathbf{x} - \mathbf{t}_i\|). \tag{5.69}$$

Для данного разложения (5.69) функции аппроксимации $F^*(\mathbf{x})$ задача сводится к определению нового множества весов $\{w_i | \quad i = 1, 2, \ldots, m_1\}$, которое минимизирует новый функционал стоимости $\mathbf{E}(F^*)$, определяемый следующим образом:

$$\mathbf{E}(F^*) = \sum_{i=1}^{N} \left(d_i - \sum_{j=1}^{m_1} w_j G(\|\mathbf{x}_i - \mathbf{t}_j\|) \right)^2 + \lambda \|\mathbf{D}F^*\|^2. \tag{5.70}$$

Первое слагаемое в правой части (5.70) может быть выражено как квадратичная Евклидова норма $||\mathbf{d} - \mathbf{G}\mathbf{w}||^2$, где

$$\mathbf{d} = [d_1, d_2, \ldots, d_N]^T, \tag{5.71}$$

$$\mathbf{G} = \begin{bmatrix} G(\mathbf{x}_1, \mathbf{t}_1) & G(\mathbf{x}_1, \mathbf{t}_2) & \ldots & G(\mathbf{x}_1, \mathbf{t}_{m_1}) \\ G(\mathbf{x}_2, \mathbf{t}_1) & G(\mathbf{x}_2, \mathbf{t}_2) & \ldots & G(\mathbf{x}_2, \mathbf{t}_{m_1}) \\ \ldots & \ldots & \ldots & \ldots \\ G(\mathbf{x}_N, \mathbf{t}_1) & G(\mathbf{x}_N, \mathbf{t}_2) & \ldots & G(\mathbf{x}_N, \mathbf{t}_{m_1}) \end{bmatrix}, \tag{5.72}$$

$$\mathbf{w} = [w_1, w_2, \ldots, w_{m_1}]^T. \tag{5.73}$$

Вектор желаемого отклика остается, как и ранее, N-мерным. Однако размерности матрицы $\mathbf{G}$ функций Грина и вектора весов $\mathbf{w}$ изменяются. Матрица $\mathbf{G}$ теперь имеет размер $N \times m_1$ и, таким образом, перестает быть симметричной. Вектор $\mathbf{w}$ имеет размерность $m_1 \times 1$. Из выражения (5.69) видно, что функция аппроксимации F^* является линейной комбинацией функций Грина для стабилизатора $\mathbf{D}$. Следовательно,

второе слагаемое в правой части (5.70) можно выразить следующим образом:

$$\begin{aligned}
&\|\mathbf{D}F^*\|^2 = (\mathbf{D}F^*, \mathbf{D}F^*)_{\mathbf{H}} = \\
&= \left[\sum_{i=1}^{m_1} w_i G(\mathbf{x}, \mathbf{t}_i), \tilde{\mathbf{D}}\mathbf{D}\sum_{i=1}^{m_1} w_i G(\mathbf{x}, \mathbf{t}_i)\right]_{\mathbf{H}} = \\
&= \left[\sum_{i=1}^{m_1} w_i G(\mathbf{x}, \mathbf{t}_i), \sum_{i=1}^{m_1} w_i \boldsymbol{\delta}\, \mathbf{t}_i)\right]_{\mathbf{H}} = \\
&= \sum_{j=1}^{m_1}\sum_{i=1}^{m_1} w_j w_i G(\mathbf{t}_j, \mathbf{t}_i) = \\
&= \mathbf{w}^T \mathbf{G}_0 \mathbf{w},
\end{aligned} \tag{5.74}$$

где во второй и в третьей строках используется определение сопряженного оператора и выражение (5.35) соответственно. Матрица $\mathbf{G}_0$ является симметричной матрицей размерности $m_1 \times m_1$:

$$\mathbf{G}_0 = \begin{bmatrix} G(\mathbf{t}_1, \mathbf{t}_1) & G(\mathbf{t}_1, \mathbf{t}_2) & \ldots G(\mathbf{t}_1, \mathbf{t}_{m_1}) \\ G(\mathbf{t}_2, \mathbf{t}_1) & G(\mathbf{t}_2, \mathbf{t}_2) & \ldots G(\mathbf{t}_2, \mathbf{t}_{m_1}) \\ \ldots & \ldots & \ldots\ldots \\ G(\mathbf{t}_{m_1}, \mathbf{t}_1) & G(\mathbf{t}_{m_1}, \mathbf{t}_2) & \ldots G(\mathbf{t}_{m_1}, \mathbf{t}_{m_1}) \end{bmatrix}. \tag{5.75}$$

Таким образом, минимизация (5.70) по отношению к вектору $\mathbf{w}$ приводит к следующему результату (см. задачу 5.5):

$$(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{G}_0)\mathbf{w} = \mathbf{G}^T\mathbf{d}. \tag{5.76}$$

Если параметр регуляризации λ принимает значение, равное нулю, вектор $\mathbf{w}$ сходится к псевдообратному (по минимальной норме) решению *сверхопределенной* (overdetermined) задачи подбора кривой на основе *метода наименьших квадратов* (least-squares data-fitting problem) для $m_1 < N$ [160]:

$$\mathbf{w} = \mathbf{G}^+\mathbf{d}, \lambda = 0, \tag{5.77}$$

где $\mathbf{G}^+$ — матрица, псевдообратная матрице $\mathbf{G}$, т.е.

$$\mathbf{G}^+ = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T. \tag{5.78}$$

Взвешенная норма

Норма в приближенном решении (5.69) обычно рассматривается как Евклидова. Однако если отдельные элементы входного вектора $\mathbf{x}$ принадлежат разным классам, более целесообразным будет использование общей *взвешенной нормы* (weighted norm) —

квадратичной формы, имеющей вид [847]

$$\|\mathbf{x}\|_C^2 = (\mathbf{Cx})^T(\mathbf{Cx}) = \mathbf{x}^T\mathbf{C}^T\mathbf{CX}, \tag{5.79}$$

где $\mathbf{C}$ — *матрица взвешенной нормы* (norm weight matrix) размерности $m_0 \times m_0$; m_0 — размерность входного вектора $\mathbf{x}$.

Используя определение взвешенной нормы, выражение для аппроксимации регуляризированного решения (5.69) можно переписать в более общей форме [675], [847]:

$$F^*(\mathbf{x}) = \sum_{i=1}^{m_1} w_i G(\|\mathbf{x} - \mathbf{t}_i\|_C). \tag{5.80}$$

Использование взвешенной нормы можно интерпретировать двумя способами. Во-первых, как применение *аффинного преобразования* (affine transformation) к исходному входному пространству. В принципе, возможность такого преобразования не искажает результаты, так как оно фактически соответствует единичной матрице взвешивания нормы. Во-вторых, взвешенная норма вытекает непосредственно из незначительного обобщения m_0-мерного Лапласиана в определении псевдодифференциального оператора (5.63) (см. задачу 5.6). Использование взвешенной нормы может быть обосновано в контексте гауссовых радиальных базисных функций следующим образом. Гауссова радиальная базисная функция $G(\|\mathbf{x}\text{-}\mathbf{t}_i\|_C)$ с центром в точке $\mathbf{t}_i$ и матрицей взвешивания нормы $\mathbf{C}$ может быть представлена следующим образом:

$$\begin{aligned} G(\|\mathbf{x} - \mathbf{t}_i\|_C) &= \exp[-(\mathbf{x} - \mathbf{t}_i)^T\mathbf{C}^T\mathbf{C}(\mathbf{x} - \mathbf{t}_i)] = \\ &= \exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{t}_i)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \mathbf{t}_i)\right], \end{aligned} \tag{5.81}$$

где обратная матрица $\boldsymbol{\Sigma}^{-1}$ определяется соотношением

$$\frac{1}{2}\boldsymbol{\Sigma}^{-1} = \mathbf{C}^T\mathbf{C}. \tag{5.82}$$

Выражение (5.81) представляет собой многомерное распределение Гаусса с вектором среднего значения $\mathbf{t}_i$ и матрицей ковариации $\boldsymbol{\Sigma}$, т.е. обобщение распределения, описываемого формулой (5.59).

Решение задачи аппроксимации, представленное формулой (5.70), открывает возможность для создания *обобщенных сетей на основе радиальных базисных функций* (RBF), имеющих структуру, показанную на рис. 5.5. Конструкция этой сети предусматривает использование порога (т.е. независимой от данных переменной), применяемого к выходному узлу сети. Это делается путем установки одного из линейных весов выходного слоя сети равным порогу и назначения соответствующей радиальной базисной функции постоянного значения $+1$.

В структурных терминах обобщенная сеть RBF на рис. 5.5 аналогична RBF-сети регуляризации, представленной на рис. 5.4. Однако они имеют два важных отличия.

1. Количество улов скрытого слоя в обобщенной сети RBF равно m_1, где m_1 обычно меньше количества примеров обученияN. С другой стороны, количество скрытых узлов в RBF-сетях регуляризации строго равно N.
2. В обобщенных сетях RBF (см. рис. 5.5) линейные веса, связанные с выходным слоем, положение центров радиальных базисных функций и матрица взвешенной нормы, соответствующая скрытому слою, являются неизвестными параметрами, которые определяются в процессе обучения. В функции активации скрытого слоя в RBF-сетях регуляризации (см. рис. 5.4) известны, так как определяются множеством функций Грина с центрами в точках примеров обучения. Единственными неизвестными параметрами этой сети являются линейные веса выходного слоя.

Рецептивные поля

Матрица ковариации $\mathbf{\Sigma}$ определяет *рецептивное поле*, или *поле чувствительности* (receptive field), гауссовой радиальной базисной функции $G(||\mathbf{x} - \mathbf{t}_i||_C)$, определенной формулой (5.81). Для известного центра $\mathbf{t}_i$ поле чувствительности $G(||\mathbf{x} - \mathbf{t}_i||_C)$ обычно определяется как опорная функция

$$\Psi(\mathbf{x}) = G(\|\mathbf{x} - \mathbf{t}_i\|_C) - a, \tag{5.83}$$

где a — некоторая положительная константа [1169]. Другими словами, поле чувствительности функции $G(||\mathbf{x}\text{-}\mathbf{t}_i||_C)$ является подмножеством области определения входного вектора $\mathbf{x}$, в которой эта функция принимает особенно большие значения (больше заданного уровня a).

Аналогично определению матрицы взвешенной нормы $\mathbf{C}$ можно выделить три различных способа задания матрицы ковариации σ, отражающие ее влияние на форму, размеры и ориентацию поля чувствительности.

1. $\mathbf{\Sigma} = \sigma^2\mathbf{I}$, где $\mathbf{I}$ — единичная матрица; σ^2 — общая дисперсия. В этом случае поле чувствительности функции $G(||\mathbf{x} - \mathbf{t}_i||_C)$ представляет собой гиперсферу с центром $\mathbf{t}_i$ и радиусом σ.
2. $\mathbf{\Sigma} = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_{m_0}^2)$, где σ_j^2 — дисперсия j-го элемента входного вектора $\mathbf{x}$; $j = 1, 2, \ldots, m_0$. В этом случае поле чувствительности функции $G(||\mathbf{x} - \mathbf{t}_i||_C)$ представляет собой гиперэллипсоид, полуоси которого совпадают с осями входного пространства и имеют длину σ_j.
3. $\mathbf{\Sigma}$ — недиагональная матрица. По определению матрица ковариации $\boldsymbol{\sigma}$ является положительно-определенной. Исходя из этого, можно использовать

преобразование подобия из алгебры матриц для декомпозиции матрицы $\boldsymbol{\sigma}$ следующим образом:

$$\boldsymbol{\Sigma} = \mathbf{Q}^T \boldsymbol{\Lambda} \mathbf{Q}, \tag{5.84}$$

где $\boldsymbol{\Lambda}$ — диагональная матрица; $\mathbf{Q}$ — матрица ортогонального вращения. Матрица $\boldsymbol{\Lambda}$ определяет форму и размер поля восприятия, а матрица $\mathbf{Q}$ — его ориентацию.

5.8. Задача XOR (повторное рассмотрение)

Рассмотрим еще раз задачу XOR (исключающего ИЛИ), которая уже была решена в главе 4 с помощью многослойного персептрона с одним скрытым слоем. В этом разделе приводится решение этой же задачи, но уже с помощью сетей RBF.

Исследуемая сеть RBF состоит из пары функций Гаусса вида

$$G(||\mathbf{x} - \mathbf{t}_i||) = \exp(-||\mathbf{x} - \mathbf{t}_i||^2), i = 1, 2, \tag{5.85}$$

с центрами $\mathbf{t}_1$ и $\mathbf{t}_2$, определяемыми выражением

$$\mathbf{t}_1 = [1, 1]^T,$$
$$\mathbf{t}_2 = [0, 0]^T.$$

Для описания выходного элемента введем следующие предположения.

1. Функционирование выходного элемента основано на *совместном использовании весов* (weight-sharing), что обусловлено симметричностью задачи. Это одна из форм встраивания априорной информации в конструкцию сети. Таким образом, при наличии двух скрытых элементов нам придется определить всего один вес w.
2. Выходной элемент имеет порог b (независимую от данных переменную). Важность этого порога объясняется тем, что функция XOR имеет среднее значение, отличное от нуля.

Описываемая таким образом структура сети, предназначенной для решения задачи XOR, показана на рис. 5.6. Соотношение "вход-выход" для этой сети имеет следующий вид:

$$y(\mathbf{x}) = \sum_{i=1}^{2} wG(||\mathbf{x} - \mathbf{t}_i||) + b. \tag{5.86}$$

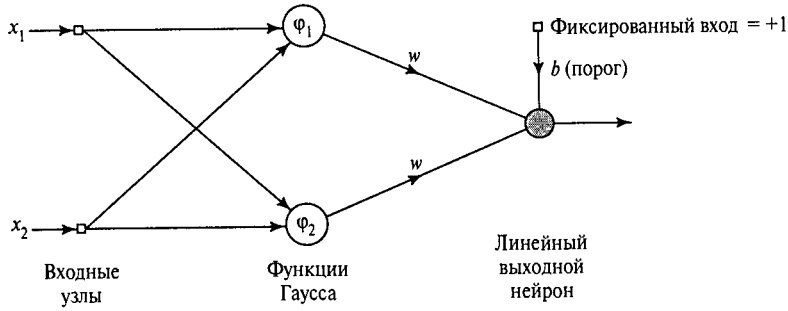


Рис. 5.6. RBF-сеть, созданная для решения задачи XOR

В соответствии с табл. 5.2 выход сети должен удовлетворять следующему требованию:

$$y(\mathbf{x}_j) = d_j, j = 1, 2, 3, 4, \tag{5.87}$$

где $\mathbf{x}_j$ — входной вектор; d_j — соответствующее ему значение желаемого отклика. Пусть

$$g_{ji} = G(||\mathbf{x}_j - \mathbf{t}_i||), j = 1, 2, 3, 4; i = 1, 2. \tag{5.88}$$

Тогда, используя значения из табл. 5.2 в выражении (5.88), можно получить следующую систему уравнений, записанную в матричном виде:

$$\mathbf{Gw} = \mathbf{d}, \tag{5.89}$$

где

$$\mathbf{G} = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1 & 1 \\ 0.3678 & 0.3678 & 1 \end{bmatrix}, \tag{5.90}$$

$$\mathbf{d} = [0101]^T, \tag{5.91}$$

$$\mathbf{w} = [wwb]^T. \tag{5.92}$$

Описанная здесь задача является *сверхопределенной* (overdeterminated) в том смысле, что количество точек данных превышает число свободных параметров. Это объясняет неквадратную форму матрицы $\mathbf{G}$. Следовательно, обратная ей матрица определяется неоднозначно. Чтобы обойти эту трудность, мы будем использовать *минимальное по норме* (minimum-norm) решение уравнения (5.77):

$$\mathbf{w} = \mathbf{G}^+\mathbf{d} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{d}. \tag{5.93}$$

ТАБЛИЦА 5.2. Преобразование "вход-выход", вычисляемое для задачи XOR

Точка данных, j	*Входной пример, $\mathbf{x}_j$*	*Желаемый отклик, d_j*
1	(1,1)	0
2	(0,1)	1
3	(0,0)	0
4	(1,0)	1

Обратите внимание, что $\mathbf{G}^T\mathbf{G}$ — квадратная матрица, для которой существует единственная обратная ей матрица. Подставляя (5.90) в (5.93), получим:

$$\mathbf{G}^+ = \begin{bmatrix} 1.8292 & -1.2509 & 0.6727 & -1.2509 \\ 0.6727 & -1.2509 & 1.8292 & -1.2509 \\ -0.9202 & 1.4202 & -0.9202 & 1.4202 \end{bmatrix}. \tag{5.94}$$

В заключение, подставляя (5.91) и (5.94) в (5.93), получим:

$$\mathbf{w} = \begin{bmatrix} -2.5018 \\ -2.5018 \\ +2.8404 \end{bmatrix},$$

что и завершает определение архитектуры сети RBF.

5.9. Оценивание параметра регуляризации

Параметр регуляризации λ играет важную роль в теории регуляризации сетей на основе радиальных базисных функций, которая рассматривалась в разделах 5.5–5.7. Для того чтобы сполна использовать возможности этой теории, необходимо применить столь же принципиальный подход к оценке параметра λ.

Чтобы лучше понять идею, рассмотрим *задачу нелинейной регрессии* (nonlinear regression problem), описываемую моделью, в которой наблюдаемый выход y_i в момент времени i зависит от входного вектора $\mathbf{x}_i$:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \tag{5.95}$$

где $f(\mathbf{x}_i)$ — гладкая кривая; ε_i — одна из реализаций процесса белого шума с нулевым средним значением и дисперсией σ^2, т.е.

$$E(\varepsilon_i) = 0 \text{ для всех } i, \tag{5.96}$$

$$E[\varepsilon_i \varepsilon_k] = \begin{cases} \sigma^2, & k = i, \\ 0, & k \neq i. \end{cases} \tag{5.97}$$

Задача состоит в восстановлении функции модели $f(\mathbf{x}_i)$ на основе данных обучения $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$.

Пусть $F_\lambda(\mathbf{x})$ — регуляризированная оценка функции $f(\mathbf{x})$ для некоторого значения параметра регуляризации λ. Это значит, что $F_\lambda(\mathbf{x})$ минимизирует функционал Тихонова, построенный для задачи нелинейной регрессии в следующем виде:

$$\mathbf{E}(F) = \frac{1}{2}\sum_{i=1}^{N}[y_i - F(\mathbf{x}_i)]^2 + \frac{\lambda}{2}\|\mathbf{D}F(\mathbf{x})\|^2. \tag{5.98}$$

Выбор подходящего значения параметра λ является нетривиальной задачей. Значение этого параметра должно обеспечивать компромисс между следующими двумя противоречивыми моментами.

- *Грубость* (roughness) решения, определяемая слагаемым $\|\mathbf{DF}(\mathbf{x})\|^2$.
- *Недостоверность* (infidelity) данных, определяемая слагаемым $\sum_{i=1}^{N}[y_i - F(\mathbf{x}_i)]^2$.

Хороший выбор параметра регуляризации λ и станет главной темой настоящего раздела.

Среднеквадратическая ошибка

Пусть $R(\lambda)$ — среднеквадратическая ошибка на данном множестве примеров между двумя функциями: функцией регрессии $f(\mathbf{x})$, соответствующей модели, и функцией аппроксимации $F_\lambda(\mathbf{x})$, представляющей решение для некоторого значения λ, т.е.

$$R(\lambda) = \frac{1}{N}\sum_{i=1}^{N}[f(\mathbf{x}_i) - F_\lambda(\mathbf{x}_i)]^2. \tag{5.99}$$

Оптимальным считается такое значение параметра λ, которое минимизирует функционал $R(\lambda)$.

Пусть $F_\lambda(\mathbf{x}_k)$ — линейная комбинация данного множества наблюдений:

$$F_\lambda(\mathbf{x}_k) = \sum_{i=1}^{N} a_{ki}(\lambda) y_i. \tag{5.100}$$

В матричном виде можно записать эквивалентное выражение

$$\mathbf{F}_\lambda = \mathbf{A}(\lambda)\mathbf{y}, \tag{5.101}$$

где

$$\mathbf{F}_\lambda = [F_\lambda(\mathbf{x}_1), F_\lambda(\mathbf{x}_2), \ldots, F_\lambda(\mathbf{x}_N)]^T,$$
$$\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$$

и

$$\mathbf{A}(\lambda) = \begin{bmatrix} a_{11} & a_{12} & \ldots a_{1N} \\ a_{21} & a_{22} & \ldots a_{2N} \\ \ldots & \ldots & \ldots\ldots \\ a_{N1} & a_{N2} & \ldots a_{NN} \end{bmatrix}. \tag{5.102}$$

Матрица $\mathbf{A}(\lambda)$ размерности $N \times N$ называется *матрицей влияний* (influence matrix).

Используя матричные обозначения, выражение (5.99) можно переписать в следующем виде:

$$R(\lambda) = \frac{1}{N} \|\mathbf{f} - \mathbf{F}_\lambda\|^2 = \frac{1}{N} \|\mathbf{f} - \mathbf{A}(\lambda)\mathbf{y}\|^2, \tag{5.103}$$

где $\mathbf{f}$ — следующий вектор размерности $N \times 1$:

$$\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)]^T.$$

Теперь можно продвинуться еще на один шаг в матричных выкладках, переписав уравнение (5.95) в виде

$$\mathbf{y} = \mathbf{f} + \boldsymbol{\varepsilon}, \tag{5.104}$$

где

$$\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N]^T.$$

Подставляя (5.104) в (5.103) и раскрывая квадрат разности, получим:

$$\begin{aligned} R(\lambda) &= \frac{1}{N} \|(\mathbf{I} - \mathbf{A}(\lambda))\mathbf{f} - \mathbf{A}(\lambda)\boldsymbol{\varepsilon}\|^2 = \\ &= \frac{1}{N} \|(\mathbf{I} - \mathbf{A}(\lambda))\mathbf{f}\|^2 - \frac{2}{N}\boldsymbol{\varepsilon}^T\mathbf{A}(\lambda)(\mathbf{I} - \mathbf{A}(\lambda))\mathbf{f} + \frac{1}{N} \|\mathbf{A}(\lambda)\boldsymbol{\varepsilon}\|^2, \end{aligned} \tag{5.105}$$

где $\mathbf{I}$ — единичная матрица размерности $N \times N$.

Для того чтобы найти искомое значение $R(\lambda)$, необходимо обратить внимание на следующие вопросы.

- Первое слагаемое в правой части (5.105) является константой, и, таким образом, на него не влияет оператор математического ожидания.
- Математическое ожидание второго слагаемого равно нулю, что следует из (5.96).
- Матожидание скаляра $||\mathbf{A}(\lambda)\boldsymbol{\varepsilon}||^2$ равно

$$E[||\mathbf{A}(\lambda)\boldsymbol{\varepsilon}||^2] = E[\boldsymbol{\varepsilon}^T\mathbf{A}^T(\lambda)\mathbf{A}(\lambda)\boldsymbol{\varepsilon}] = \\ = \mathrm{tr}\{E[\boldsymbol{\varepsilon}^T\mathbf{A}^T(\lambda)\mathbf{A}(\lambda)\boldsymbol{\varepsilon}]\} = E\{\mathrm{tr}[\boldsymbol{\varepsilon}^T\mathbf{A}^T(\lambda)\mathbf{A}(\lambda)\boldsymbol{\varepsilon}]\}. \tag{5.106}$$

При вычислении этого значения сначала используется тот факт, что след скаляра равен ему самому, после чего изменяется порядок операторов математического ожидания и следа.

Далее будем использовать правило, взятое из алгебры матриц: для данных двух матриц совместимого размера $\mathbf{B}$ и $\mathbf{C}$ след произведения $\mathbf{BC}$ равен следу произведения $\mathbf{CB}$. Таким образом, принимая $\mathbf{B} = \boldsymbol{\varepsilon}^T$ и $\mathbf{C} = \mathbf{A}^T(\lambda)\mathbf{A}(\lambda)\boldsymbol{\varepsilon}$, выражение (5.106) можно переписать в эквивалентной форме:

$$E[||\mathbf{A}(\lambda)\mathbf{f}||^2] = E\{\mathrm{tr}[\mathbf{A}^T(\lambda)\mathbf{A}(\lambda)\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T]\} = \sigma^2\mathrm{tr}[\mathbf{A}^T(\lambda)\mathbf{A}(\lambda)], \tag{5.107}$$

где в последней выкладке используется соотношение (5.97). В заключение заметим, что след $\mathbf{A}^T(\lambda)\mathbf{A}(\lambda)$ равен следу $\mathbf{A}^2(\lambda)$, т.е. (5.107) можно переписать в следующем виде:

$$E[||\mathbf{A}(\lambda)\mathbf{f}||^2] = \sigma^2\mathrm{tr}[\mathbf{A}^2(\lambda)]. \tag{5.108}$$

Объединяя эти три результата, ожидаемое значение $R(\lambda)$ можно выразить в виде

$$E[R(\lambda)] = \frac{1}{N}\|\mathbf{I} - \mathbf{A}(\lambda)\mathbf{f}\|^2 + \frac{\sigma^2}{N}\mathrm{tr}[\mathbf{A}^2(\lambda)]. \tag{5.109}$$

Однако среднеквадратическая ошибка на данном множестве примеров $R(\lambda)$ не так важна с практической точки зрения, поскольку она требует априорных знаний о функции регрессии $f(\mathbf{x})$, которую требуется восстановить. Поэтому в качестве оценки $E[R(\lambda)]$ введем следующее определение [231]:

$$\hat{R}(\lambda) = \frac{1}{N}\|(\mathbf{I} - \mathbf{A}(\lambda))\mathbf{y}\|^2 + \frac{\sigma^2}{N}\mathrm{tr}[\mathbf{A}^2(\lambda)] - \frac{\sigma^2}{N}\mathrm{tr}[(\mathbf{I} - \mathbf{A}(\lambda))^2]. \tag{5.110}$$

Эта оценка является *несмещенной* (unbiased), а значит (следуя процедуре, аналогичной выводу выражения (5.109)), можно показать, что

$$E[\hat{R}(\lambda)] = E[R(\lambda)]. \tag{5.111}$$

Следовательно, выбор подходящего значения параметра регуляризации λ обеспечивается путем минимизации оценки $\hat{R}(\lambda)$.

Обобщенная перекрестная проверка

Недостатком оценки $\hat{R}(\lambda)$ является то, что она требует знания дисперсии шума σ^2. В ситуациях, встречающихся на практике, этот параметр обычно неизвестен. Для того чтобы обеспечить работоспособность алгоритма, в подобных случаях обычно используется концепция обобщенной перекрестной проверки, которая была выдвинута в [231].

Сначала адаптируем к нашей задаче обычную форму перекрестной проверки (см. главу 4). В частности, пусть $F_\lambda^{[k]}$ минимизирует функционал

$$\mathbf{E}(F) = \frac{1}{2}\sum_{i=1,i}^{N}[y_i - F_\lambda(\mathbf{x}_i)]^2 + \frac{\lambda}{2}\|\mathbf{D}F(\mathbf{x})\|^2, \tag{5.112}$$

где k-й член $[y_i - F_\lambda(\mathbf{x}_k)]$ вынесен из слагаемого стандартной ошибки для получения возможности "прогнозировать" отсутствующие точки данных y_k, проверяя, таким образом, качество оценки параметра λ. Следовательно, можно ввести следующий показатель качества:

$$V_0(\lambda) = \frac{1}{2}\sum_{k=1}^{N}[y_k - F_\lambda^{[k]}(\mathbf{x}_k)]^2, \tag{5.113}$$

который зависит только от самих данных. Таким образом, *обычная перекрестная оценка* (ordinary cross-validation estimate) параметра λ определяется как аргминимум функции $V_0(\lambda)$ [1105].

Отметим одно полезное свойство $F_\lambda^{[k]}(\mathbf{x}_k)$. Если точку y_k заменить ее оценкой $F_\lambda^{[k]}(\mathbf{x}_k)$, а исходный функционал Тихонова $\mathbf{E}(F)$ из (5.98) достигает минимума в точке $y_0, \ldots, y_{k-1}, y_k, y_{k+1}, \ldots, y_N$, то $F_\lambda^{[k]}(\mathbf{x}_k)$ является решением. Это свойство, наряду с тем фактом, что для каждого входного вектора $\mathbf{x}$ аргминимум $F_\lambda(\mathbf{x})$ функционала $\mathbf{E}(F)$ линейно зависит от y_k, позволяет записать следующее:

$$F_\lambda^{[k]}(\mathbf{x}_k) = F_\lambda(\mathbf{x}_k) + (F_\lambda^{[k]}(\mathbf{x}_k) - y_k)\frac{\partial F_\lambda(\mathbf{x}_k)}{\partial y_k}. \tag{5.114}$$

Из определения (5.100), задающего элементы матрицы влияний $\mathbf{A}(\lambda)$, видно, что

$$\frac{\partial F_\lambda(\mathbf{x}_k)}{\partial y_k} = a_{kk}(\lambda), \tag{5.115}$$

где a_{kk} — k-й диагональный элемент матрицы $\mathbf{A}(\lambda)$. Отсюда, подставляя (5.115) в (5.114) и разрешая результат относительно $F_\lambda^{[k]}(\mathbf{x}_k)$, получим:

$$F_\lambda^{[k]}(\mathbf{x}_k) = \frac{F_\lambda(\mathbf{x}_k) - a_{kk}(\lambda)y_k}{1 - a_{kk}(\lambda)} = \frac{F_\lambda(\mathbf{x}_k) - y_k}{1 - a_{kk}(\lambda)} + y_k. \tag{5.116}$$

Подставляя выражение (5.116) в (5.113), можно переопределить функцию $V_0(\lambda)$ следующим образом:

$$V_0(\lambda) = \frac{1}{N}\sum_{k=1}^{N}\left[\frac{y_k - F_\lambda(\mathbf{x}_k)}{1 - a_{kk}(\lambda)}\right]^2. \tag{5.117}$$

Обычно значения $a_{kk}(\lambda)$ отличаются для каждого k, а это значит, что точки данных в $V_0(\lambda)$ нельзя трактовать как равноценные. Чтобы обойти это нежелательное свойство обычной перекрестной проверки, в [231] была введена процедура *обобщенной перекрестной проверки* (generalized cross validation или GVC), использующая поворот координат[11]. В частности, функция обычной перекрестной проверки $V_0(\lambda)$

---

[11] Для того чтобы вывести функцию обобщенной перекрестной проверки из обычной, можно рассмотреть задачу гребневой регрессии (ridge regression problem), описанную в [1105]:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\alpha} + \boldsymbol{\varepsilon}, \tag{1}$$

где $\mathbf{X}$ — матрица входных сигналов размерности $N \times N$, а вектор $\boldsymbol{\varepsilon}$ имеет среднее значение, равное нулю, и матрицу ковариации, равную $\sigma^2\mathbf{I}$. Используя сингулярную декомпозицию значений $\mathbf{X}$, можно записать следующее:

$$\mathbf{X} = \mathbf{UDV}^T,$$

где $\mathbf{U}$ и $\mathbf{V}$ — ортогональные матрицы; $\mathbf{D}$ — диагональная матрица. Пусть $\tilde{\mathbf{y}} = \mathbf{U}^T\mathbf{y}$, $\beta = \mathbf{V}^T\alpha$, $\tilde{\varepsilon} = \mathbf{U}^T\varepsilon$.

Тогда для преобразования выражения (1) можно использовать матрицы $\mathbf{U}$ и $\mathbf{V}$:

$$\tilde{\mathbf{y}} = \mathbf{D}\beta + \tilde{\varepsilon}. \tag{2}$$

Диагональная матрица $\mathbf{D}$ (не следует путать это обозначение с оператором дифференцирования) выбирается так, чтобы ее сингулярные значения составляли пары. Тогда существует диагональная матрица $\mathbf{W}$, такая, что $\mathbf{WDW}^T$ является циркулянтной матрицей, а именно: $\mathbf{A} = \mathbf{WDW}^T = \begin{bmatrix} a_0 & a_1 & \dots & a_{N-1} \\ a_{N-1} & a_0 & \dots & a_{N-2} \\ a_{N-2} & a_{N-1} & \dots & a_{N-3} \\ \dots & \dots & \dots & \dots \\ a_1 & a_2 & \dots & a_0 \end{bmatrix}$,

в которой элементы, расположенные по каждой из диагоналей, одинаковы. Пусть $\mathbf{z} = \mathbf{W}\tilde{\mathbf{y}}$, $\gamma = \mathbf{W}\beta$, $\boldsymbol{\xi} = \mathbf{W}\tilde{\varepsilon}$.

Тогда для преобразования выражения (2) можно использовать матрицу $\mathbf{W}$.

$$\mathbf{z} = \mathbf{A}\gamma + \boldsymbol{\xi}. \tag{3}$$

Диагональная матрица $\mathbf{D}$ имеет "максимально несвязанные" строки, в то время как строки матрицы $\mathbf{A}$ "максимально объединены".

Выполнив такое преобразование, можно утверждать, что обобщенная перекрестная проверка является эквивалентом преобразования задачи гребневой регрессии (1) к виду (3) с последующим обратным преобразованием в обычную систему координиат [1105].

(5.117) была изменена следующим образом:

$$V(\lambda) = \frac{1}{N}\sum_{k=1}^{N}\omega_k\left[\frac{y_k - F_\lambda(\mathbf{x}_k)}{1 - a_{kk}(\lambda)}\right]^2, \tag{5.118}$$

где сами веса ω_k определяются следующим образом:

$$\omega_k = \left[\frac{1 - a_{kk}(\lambda)}{\frac{1}{N}\mathrm{tr}[\mathbf{I} - \mathbf{A}(\lambda)]}\right]^2. \tag{5.119}$$

Таким образом, обобщенная функция перекрестной проверки принимает вид

$$V(\lambda) = \frac{\frac{1}{N}\sum_{k=1}^{N}[y_k - F_\lambda(\mathbf{x}_k)]^2}{\left[\frac{1}{N}\mathrm{tr}[\mathbf{I} - \mathbf{A}(\lambda)]\right]^2}. \tag{5.120}$$

В заключение, подставляя (5.100) в (5.120), получим:

$$V(\lambda) = \frac{\frac{1}{N}\|\mathbf{I} - \mathbf{A}(\lambda)\mathbf{y}\|^2}{\left[\frac{1}{N}\mathrm{tr}[\mathbf{I} - \mathbf{A}(\lambda)]\right]^2}. \tag{5.121}$$

Эта формула включает величины, зависящие только от данных.

Оптимальное свойство обобщенной функции перекрестной проверки V(λ)

Пусть λ минимизирует ожидаемое значение обобщенной функции перекрестной проверки$V(\lambda)$. *Неэффективность математического ожидания* (expectation inefficiency) в методе обобщенной перекрестной проверки определяется выражением

$$I^* = \frac{E[R(\lambda)]}{\min\limits_{\lambda} E[R(\lambda)]}, \tag{5.122}$$

где $R(\lambda)$ — среднеквадратическая ошибка на множестве данных, определяемая формулой (5.99). Обычно асимптотическое значение I^* удовлетворяет следующему условию:

$$\lim_{N\to\infty} I^* = 1. \tag{5.123}$$

Другими словами, при больших N среднеквадратическая ошибка $R(\lambda)$ по λ, вычисленная с помощью минимизации обобщенной функции перекрестной проверки $V(\lambda)$, приближается к минимально возможному значению $R(\lambda)$. Это обеспечивает применимость $V(\lambda)$ для оценивания параметра λ.

Заключительные комментарии

Общая идея выбора параметра регуляризации λ состоит в минимизации среднеквадратической ошибки на множестве данных обучения — $R(\lambda)$. К сожалению, эту задачу нельзя решить напрямую, так как выражение для $R(\lambda)$ содержит неизвестную функцию регрессии $f(\mathbf{x})$. В связи с этим существуют два метода, которые можно применить на практике.

- Если известна дисперсия шума σ^2, то в качестве оптимального значения λ можно использовать аргминимум оценки $\hat{R}(\lambda)$ из (5.110) (здесь под оптимальностью понимается то, что данное значение λ также минимизирует и $R(\lambda)$).
- Если дисперсия шума σ^2 неизвестна, в качестве оптимального значения параметра λ можно использовать аргминимум обобщенной функции перекрестной проверки $V(\lambda)$ из (5.121). При таком подходе обеспечивается минимально возможное значение среднеквадратической ошибки при $N \to \infty$.

Важно отметить, что теория, обосновывающая использование обобщенной функции перекрестной проверки для оценки параметра λ, является асимптотической, т.е. хороших результатов можно добиться лишь тогда, когда множество данных достаточно велико для того, чтобы отличить полезный сигнал от шума.

Практический опыт работы с обобщенной функцией перекрестной проверки показал, что этот метод робастен относительно неоднородности дисперсии и негауссова шума [1105]. Однако если шум представляет собой хорошо коррелированный процесс, метод дает неудовлетворительную оценку параметра регуляризации λ.

В заключение хотелось бы привести некоторые комментарии относительно вычисления обобщенной функции перекрестной проверки $V(\lambda)$. Для данных пробных значений параметра регуляризации λ нахождение знаменателя $[\mathrm{tr}[\mathbf{I} - \mathbf{A}(\lambda)]/N]^2$ в формуле (5.121) является самой затратной частью работы по вычислению функции $V(\lambda)$. Для вычисления следа матрицы $\mathbf{A}(\lambda)$ можно использовать метод *рандомизированного вычисления* следа (randomized trace) описанный в [1106]. Применение этого метода в особо больших системах будет оправданным.

5.10. Свойства аппроксимации сетей RBF

В главе 4 уже рассматривались свойства аппроксимации многослойного персептрона. Сети на основе радиальных базисных функций также демонстрируют хорошие свойства аппроксимации. Семейство сетей RBF является достаточно широким, чтобы равномерно аппроксимировать любую непрерывную функцию на компактном множестве[12].

[12] В приложении к [853] содержится благодарность за результат, полученный в 1981 году А. Брауном (A.C.Brown), согласно которому сети RBF позволяют отобразить произвольную функцию из замкнутой области в $\Re^{m_0}$ в пространство $\Re$.

Универсальная теорема об аппроксимации

Пусть $G : \Re^{m_0} \rightarrow \Re$ — ограниченная, непрерывная и интегрируемая функция, такая, что

$$\int_{\Re^{m_0}} G(\mathbf{x})d\mathbf{x} \neq 0.$$

Пусть $\mathbf{G}_G$ — семейство сетей RBF, включающих функции $F : \Re^{m_0} \rightarrow \Re$ следующего вида:

$$F(\mathbf{x}) = \sum_{i=1}^{m_1} w_i G\left(\frac{\mathbf{x} - \mathbf{t}_i}{\sigma}\right),$$

где σ>0, $w_i \in \Re$ и $\mathbf{t}_i \in \Re^{m_0}$ для $i = 1, 2, \ldots, m_1$. Тогда выполняется следующая *теорема об универсальной аппроксимации* (universal approximation theorem) для сетей RBF.

Для любой непрерывной функции $f(\mathbf{x})$ найдется сеть RBF с множеством центров $\{\mathbf{t}_i\}_{i=1}^{m_1}$ и общей шириной $\sigma > 0$, такая, что функция $F(\mathbf{x})$, реализуемая сетью, будет близка к $f(\mathbf{x})$ по норме $L_p, p \in [1, \infty]$.

Обратите внимание, что в сформулированной таким образом обобщенной теореме об аппроксимации ядро $G : \Re^{m_0} \rightarrow \Re$ не обязательно должно удовлетворять условию радиальной симметрии. Таким образом, теорема является более строгой, чем это необходимо для сетей RBF. И что более важно, она подводит теоретический базис под построение нейронных сетей на основе радиальных базисных функций с целью их практического применения.

"Проклятие размерности" (продолжение)

Помимо свойства универсальной аппроксимации сетей RBF, необходимо рассмотреть вопрос порядка аппроксимации, обеспечиваемого этими сетями. Как было сказано в главе 4, внутренняя *сложность* класса аппроксимирующих функций экспоненциально возрастает с увеличением отношения m_0/s, где m_0 — размерность входного сигнала; s — *индекс гладкости* (smoothness index), определяемый как количество ограничений, накладываемых на аппроксимирующие функции этого конкретного класса. Проблема "проклятия размерности", сформулированная Беллманом (Bellman), означает, что независимо от используемого метода аппроксимации при постоянном ин-

В [423] рассматриваются гауссовы функции и аппроксимации на компактных подмножествах $\Re^{m_0}$, которые являются выпуклыми. Показано, что сети RBF с единственным скрытым слоем гауссовых элементов являются универсальнымн аппроксиматорами. Однако самое строгое доказательство свойства универсальной аппроксимации сетей RBF представлено в [815], которая была завершена еще до выхода в свет [423].

ТАБЛИЦА 5.3. Два способа аппроксимации и соответствующие функциональные пространства с одинаковой скоростью сходимости $O(1/\sqrt{m_1})$, где m_1 — размерность скрытого пространства

Функциональное пространство	*Норма*	*Техника аппроксимации*
$\int_{\Re^{m_0}} \|\mathbf{s}\| \tilde{F}(\mathbf{s}) d\mathbf{s} < \infty$, где $\tilde{F}(\mathbf{s})$ — многомерное преобразование Фурье для функции аппроксимации $F(\mathbf{x})$	$L_2(\Omega)$	а) Многослойный персептрон: $F(\mathbf{x}) = \sum_{i=1}^{m_1} a_i \varphi(\mathbf{w}_i^T \mathbf{x} + b_i)$, где $\varphi(\cdot)$ — сигмоидальная функция активации
Пространство функций Соболева, производные которых до порядка $2m > m_0$ являются интегрируемыми	$L_2(\Re^2)$	б) RBF-сети: $F(\mathbf{x}) = \sum_{i=1}^{m_1} a_i \exp\left(-\frac{\|\mathbf{x}-\mathbf{t}_i\|^2}{2\sigma^2}\right)$

дексе гладкости s количество параметров, необходимых для достижения требуемого уровня точности функции аппроксимации, экспоненциально возрастает в зависимости от размерности входного пространства m_0. Единственный способ добиться независимости скорости сходимости от размерности входного сигнала m_0 и, таким образом, избежать "проклятия размерности" — это увеличить индекс гладкости пропорционально увеличению количества параметров в аппроксимирующей функции, что, свою очередь, ведет к увеличению ее сложности. Это продемонстрировано в табл. 5.3, которая взята из [360].

В таблице приведены ограничения на функциональные пространства, которым должны удовлетворять рассматриваемые способы аппроксимации — многослойный персептрон и RBF-сети, для того чтобы скорость сходимости не зависела от размерности входного пространства m_0. Естественно, эти условия для различных методов аппроксимации отличаются. В случае сетей RBF используется *пространство Соболева*[13] функций, производные которых до порядка $2m > m_0$ являются интегрируемыми. Другими словами, для того чтобы скорость сходимости не зависела от роста размерности, необходимо с ростом размерности входного пространства m_0 увеличивать количество производных функции аппроксимации, которые должны быть интегрируемыми. Как уже говорилось в главе 4, аналогичное условие накладывается на многослойный персептрон, однако несколько неявным способом. Проанализировав табл. 5.3, можно сделать следующий вывод.

[13] Пусть Ω — ограниченная область в пространстве $\Re^n$ с границей Г. Рассмотрим множество **G** непрерывных вещественнозначных функций, которые имеют непрерывный градиент на множестве Ω+Г. Билинейная форма $\int_\Omega (\text{grad } u : \text{grad } v + uv) d\mathbf{x}$ является допустимым скалярным произведением на множестве **G**. Замыкание **G** по норме, сгенерированной этим скалярным произведением, называется *пространством Соболева* [246]. Пространства Соболева играют важную роль в теории уравнений в частных производных и являются важным примером гильбертовых пространств.

Пространства функций аппроксимации, достижимые многослойным персептроном и сетями RBF, становятся все более ограниченными по мере роста размерности входного пространства m_0.

В результате оказывается, что "проклятие размерности" нельзя преодолеть ни с помощью нейронных сетей, будь то многослойный персептрон или сети RBF, ни при использовании других методов аналогичной природы.

Связь между сложностью обучающего множества, вычислительной сложностью и эффективностью обобщения

Обсуждение задачи аппроксимации будет неполным без упоминания того факта, что на практике не существует неограниченной выборки данных. Обычно под рукой оказывается некоторое множество примеров вполне ограниченного размера. Аналогично, не существует нейронных сетей, обладающих бесконечно большой вычислительной мощностью, — она всегда ограничена. Следовательно, существуют два момента, приводящих к ошибке обобщения в нейронных сетях, обучаемых на конечных наборах примеров и тестируемых на не встречавшихся ранее данных. Этот вопрос уже обсуждался в главе 2. Один из этих моментов, называемый *ошибкой аппроксимации* (approximation error), возникает вследствие ограниченной мощности сети, недостаточной для представления интересующей целевой функции. Другой момент, *ошибка оценивания* (estimation error), является результатом недостаточности ограниченного объема информации, содержащегося в примерах обучения. С учетом этой информации в [787] получен предел ошибки обобщения, генерируемой RBF-сетью с функциями Гаусса, который выражается в терминах размеров скрытого слоя и обучающего множества. Вывод получен для случая обучения функции регрессии в моделях вида (5.95). Функция регрессии принадлежит некоторому пространству Соболева.

Этот предел в терминологии PAC-обучения (см. главу 2) можно сформулировать следующим образом [787].

Пусть G — класс гауссовых RBF-сетей с m_0 входными и m_1 скрытыми узлами. Пусть $f(\mathbf{x})$ — функция регрессии, принадлежащая некоторому пространству Соболева. Предполагается, что множество примеров обучения $\mathbf{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$ составляется с помощью случайной выборки из регрессивной модели, основанной на функции $f(\mathbf{x})$. Тогда для любого параметра чувствительности $\delta \in (0, 1]$ ошибка обобщения, генерируемая сетью, ограничена сверху числом

$$O\left(\frac{1}{m_1}\right) + O\left(\frac{m_0 m_1}{N}\log(m_1 N) + \frac{1}{N}\log\left(\frac{1}{\delta}\right)^{1/2}\right) \tag{5.124}$$

с вероятностью, превышающей $1 - \delta$.

Из выражения (5.124) можно сделать следующие выводы.

- *Ошибка обобщения сходится к нулю только в том случае, если количество скрытых элементов m_1 возрастает медленнее, чем размер обучающей выборки N.*
- *Для фиксированного количества примеров обучения N оптимальное количество скрытых элементов m_1^* ведет себя как (см. задачу 5.11)*

$$m_1^* \propto N^{1/3}. \tag{5.125}$$

- *Сети RBF обеспечивают скорость аппроксимации порядка $O(1/m_1)$, что близко к значению, полученному в [96] для многослойного персептрона с сигмоидальной функцией активации (см. раздел 4.12).*

5.11. Сравнение сетей RBF и многослойных персептронов

Сети на основе радиальных базисных функций (RBF) и многослойный персептрон (MLP) являются примерами нелинейных многослойных сетей прямого распространения. И те и другие являются универсальными аппроксиматорами. Таким образом, неудивительно, что всегда существует сеть RBF, способная имитировать многослойный персептрон (и наоборот). Однако эти два типа сетей отличаются по некоторым важным аспектам.

1. Сети RBF (в своей основной форме) имеют один скрытый слой, в то время как многослойный персептрон может иметь большее количество скрытых слоев.
2. Обычно *вычислительные* (computational) узлы многослойного персептрона, расположенные в скрытых и выходном слоях, используют одну и ту же модель нейрона. С другой стороны, вычислительные узлы скрытого слоя сети RBF могут в корне отличаться от узлов выходного слоя и служить разным целям.
3. Скрытый слой в сетях RBF является нелинейным, в то время как выходной — линейным. В то же время скрытые и выходной слои многослойного персептрона, используемого в качестве классификатора, являются нелинейными. Если многослойный персептрон используется для решения задач нелинейной регрессии, в качестве узлов выходного слоя обычно выбираются линейные нейроны.
4. Аргумент функции активации каждого скрытого узла сети RBF представляет собой *Евклидову норму (расстояние)* между входным вектором и центром радиальной функции. В то же время аргумент функции активации каждого скрытого узла многослойного персептрона — это *скалярное произведение* входного вектора и вектора синаптических весов данного нейрона.

5. Многослойный персептрон обеспечивает *глобальную* аппроксимацию нелинейного отображения. С другой стороны, сеть RBF с помощью экспоненциально уменьшающихся локализованных нелинейностей (т.е. функций Гаусса) создает *локальную* аппроксимацию нелинейного отображения.

Это, в свою очередь, означает, что для аппроксимации нелинейного отображения с помощью многослойного персептрона может потребоваться меньшее число параметров, чем для сети RBF при одинаковой точности вычислений.

Линейные характеристики выходного слоя сети RBF означают, что такая сеть более тесно связана с персептроном Розенблатта, чем с многослойным персептроном. Тем не менее сети RBF отличаются от этого персептрона тем, что способны выполнять нелинейные преобразования входного пространства. Это было хорошо продемонстрировано на примере решения задачи XOR, которая не может быть решена ни одним линейным персептроном, но с легкостью решается сетью RBF.

5.12. Регрессия ядра и ее связь с сетями RBF

Представленная выше теория сетей RBF создавалась с прицелом на решение задач интерполяции. В настоящем разделе мы примем другую точку зрения; займемся задачей построения *регрессии ядра* (kernel regression) на основе *оценки плотности* (density estimation).

В качестве примера рассмотрим модель нелинейной регрессии (5.95), которую повторно приведем в этом разделе для сохранения последовательности рассуждений:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, i = 1, 2, \ldots, N.$$

В качестве обоснованной оценки неизвестной функции регрессии $f(\mathbf{x})$ можно выбрать среднее по наблюдениям (т.е. значениям выходного сигнала y) в окрестности точки $\mathbf{x}$. Однако для успешной реализации такого подхода локальное среднее должно быть ограничено малой окрестностью (т.е. полем чувствительности) точки $\mathbf{x}$, так как в общем случае наблюдения, соответствующие точкам, удаленным от $\mathbf{x}$, будут иметь другие средние значения. Чтобы конкретизировать этот вопрос, вспомним, что функция $f(\mathbf{x})$, равная условному среднему значений y для данного $\mathbf{x}$ (т.е. регрессия y по $\mathbf{x}$), задается в следующем виде:

$$f(\mathbf{x}) = E[y|\mathbf{x}].$$

Используя формулу для математического ожидания случайной переменной,можно записать:

$$f(\mathbf{x}) = \int_{-\infty}^{\infty} y f_Y(y|\mathbf{x}) dy, \tag{5.126}$$

где $f_Y(y|\mathbf{x})$ — функция плотности условной вероятности Y для данного $\mathbf{x}$. Из теории вероятностей известно, что

$$f_Y(y|\mathbf{x}) = \frac{f_{\mathbf{X},Y}(\mathbf{x}, y)}{f_{\mathbf{X}}(\mathbf{x})}, \tag{5.127}$$

где $f_X(\mathbf{x})$ — плотность вероятности $\mathbf{x}$; $f_{X,Y}(\mathbf{x}, y)$ — плотность совместной вероятности $\mathbf{X}$ и Y. Подставляя (5.127) в (5.126), получим следующую формулу для функции регрессии:

$$f(\mathbf{x}) = \frac{\int_{-\infty}^{\infty} y f_{\mathbf{X},Y}(\mathbf{x}, y) dy}{f_{\mathbf{X}}(\mathbf{x})}. \tag{5.128}$$

Нас интересует ситуация, в которой функция плотности условной вероятности $f_{X,Y}(\mathbf{x}, y)$ неизвестна. Имеется лишь множество примеров обучения $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. Для того чтобы оценить функцию $f_{X,Y}(\mathbf{x}, y)$, а значит и $f_X(\mathbf{x})$, можно использовать непараметрическую функцию оценивания, получившую название *функции оценки плотности Парзена–Розенблатта* (Parzen-Rozenblatt density estimator) [819], [903], [904]. Основой для описания этой функции оценивания служит *ядро* (kernel), обозначаемое как $K(\mathbf{x})$ и обладающее свойствами, сходными со свойствами функций плотности вероятности.

- Ядро $K(\mathbf{x})$ является непрерывной, ограниченной и действительнозначной функцией аргумента $\mathbf{x}$. Оно симметрично относительно начала координат, где достигает своего максимального значения.
- Общий объем, находящийся под поверхностью ядра $K(\mathbf{x})$, равен единице, т.е. для любого m-мерного вектора $\mathbf{x}$:

$$\int_{\Re^m} K(\mathbf{x}) d\mathbf{x} = 1. \tag{5.129}$$

Предполагая, что множество $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ состоит из независимых равномерно распределенных случайных векторов, можно формально определить оценку плотности Парзена–Розенблатта функции $f(\mathbf{x})$ следующим образом:

$$\hat{f}_{\mathbf{X}}(\mathbf{x}) = \frac{1}{Nh^{m_0}} \sum_{i=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \quad \mathbf{x} \in \Re^{m_0}, \tag{5.130}$$

где параметр сглаживания h является положительным числом, называемым *шириной полосы* (bandwidth), или просто *шириной* (width). (Не путайте используемый здесь параметр h с одноименным параметром, используемым в определении производной Фреше из раздела 5.5.) Важным свойством функции оценки плотности Парзена-Розенблатта является то, что она обеспечивает *состоятельную оценку* (consistent estimator)[14], т.е. при выборе $h(N)$ как функции от N, такой, что

$$\lim_{N \to \infty} h(N) = 0,$$

выполняется равенство

$$\lim_{N \to \infty} E[\hat{f}_{\mathbf{X}}(\mathbf{x})] = f_{\mathbf{X}}(\mathbf{x}).$$

Для его выполнения необходимо, чтобы точка $\mathbf{x}$ была точкой непрерывности функции $\hat{f}_{\mathbf{X}}(\mathbf{x})$.

Аналогично можно получить и функцию оценки плотности Парзена–Розенблатта для функции плотности совместной вероятности $f_{X,Y}(\mathbf{x}, y)$:

$$\hat{f}_{\mathbf{X},Y}(\mathbf{x}, y) = \frac{1}{Nh^{m_0+1}} \sum_{i=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) K\left(\frac{y - y_i}{h}\right), \quad \mathbf{x} \in \Re^{m_0}, \; y \in \Re. \tag{5.131}$$

Интегрируя $\hat{f}_{\mathbf{X},Y}(\mathbf{x}, y)$ по y, из выражения (5.130) получим $\hat{f}_{\mathbf{X}}(\mathbf{x})$, что и требовалось доказать. Более того,

$$\int_{-\infty}^{\infty} y \hat{f}_{\mathbf{X},Y}(\mathbf{x}, y) dy = \frac{1}{Nh^{m_0+1}} \sum_{i=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \int_{-\infty}^{\infty} yK\left(\frac{y - y_i}{h}\right) dy.$$

Изменяя переменную интегрирования путем подстановки $z = (y - y_i)/h$ и воспользовавшись свойством симметрии ядра $K(\cdot)$, получим следующий результат:

$$\int_{-\infty}^{\infty} y \hat{f}_{\mathbf{X},Y}(\mathbf{x}, y) dy = \frac{1}{Nh^{m_0}} \sum_{i=1}^{N} y_i K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right). \tag{5.132}$$

[14] Доказательство асимптотического свойства функции оценки плотности Парзена–Розенблатта содержится в [166] и [819].

Используя (5.132) и (5.130) в качестве оценки значений числителя и знаменателя для формулы (5.128), можно вычислить следующую оценку функции регрессии $f(\mathbf{x})$ (после приведения подобных членов):

$$F(\mathbf{x}) = \hat{f}(\mathbf{x}) = \frac{\sum_{i=1}^{N} y_i K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)}{\sum_{j=1}^{N} K\left(\frac{\mathbf{x}-\mathbf{x}_j}{h}\right)}, \tag{5.133}$$

где в знаменателе для простоты использовался индекс суммирования j вместо i. Как и в обычных сетях RBF, функция оценки регрессии ядра $F(\mathbf{x})$, определяемая формулой (5.133), является универсальным аппроксиматором.

Функцию аппроксимации $F(\mathbf{x})$ можно рассматривать с двух позиций.

1. *Функция оценки регрессии Надарайя–Ватсона* (Nadaraya-Watson regression estimator). Определим *нормализованную функцию взвешивания* (normalized weighting function):

$$W_{N,i}(\mathbf{x}) = \frac{K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)}{\sum_{j=1}^{N} K\left(\frac{\mathbf{x}-\mathbf{x}_j}{h}\right)}, \quad i = 1, 2, \ldots, N, \tag{5.134}$$

где

$$\sum_{i=1}^{N} W_{N,i}(\mathbf{x}) = 1 \quad \mathbf{x}. \tag{5.135}$$

Тогда функцию оценки регрессии ядра (5.133) можно переписать в упрощенном виде:

$$F(\mathbf{x}) = \sum_{i=1}^{N} W_{N,i}(\mathbf{x}) y_i. \tag{5.136}$$

Это не что иное, как *взвешенное среднее* (weighted average) наблюдений y. Частный случай формулы для $W_{N,i}(\mathbf{x})$, формула (5.136), был предложен учеными Надарайя (Nadaraya) и Ватсоном (Watson) в [769] и [1118]. Поэтому функцию аппроксимации (5.136) часто называют *функцией оценки регрессии Надарайя–Ватсона* (NWRE).

2. *Нормализованные RBF-сети* (normalized RBF network). В этом случае предполагается *сферическая симметрия* ядра $K(\mathbf{x})$, т.е. [601]

$$K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) = K\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|}{h}\right) \text{ для всех } i, \tag{5.137}$$

где $||\cdot||$ — Евклидова норма вектора аргумента. Соответственно *нормализованная радиальная базисная функция* имеет вид

$$\psi_N(\mathbf{x}, \mathbf{x}_i) = \frac{K\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|}{h}\right)}{\sum_{j=1}^{N} K\left(\frac{\|\mathbf{x}-\mathbf{x}_j\|}{h}\right)}, \quad i = 1, 2, \ldots, N, \tag{5.138}$$

где

$$\sum_{i=1}^{N} \psi_N(\mathbf{x}, \mathbf{x}_i) = 1 \text{ для всех } \mathbf{x}. \tag{5.139}$$

Индекс N в обозначении $\psi_N(\mathbf{x}, \mathbf{x}_i)$ указывает на использование *нормализации*.

В рассматриваемой здесь задаче регрессии "линейные веса" w_i, применяемые к базисным функциям $\psi_N(\mathbf{x}, \mathbf{x}_i)$, являются наблюдениями y_i модели регрессии для входных примеров $\mathbf{x}_i$. Таким образом, принимая

$$y_i = w_i, i = 1, 2, \ldots, N,$$

функцию аппроксимации (5.133) можно преобразовать к более общему виду:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \psi_N(\mathbf{x}, \mathbf{x}_i). \tag{5.140}$$

Уравнение (5.140) представляет собой описание *сети на основе нормализованных радиальных базисных функций* [750], [1169]. Заметим, что

$$0 \leq \psi_N(\mathbf{x}, \mathbf{x}_i) \leq 1 \text{ для всех } \mathbf{x} \text{ и } \mathbf{x}_i. \tag{5.141}$$

Следовательно, $\psi_N(\mathbf{x}, \mathbf{x}_i)$ можно интерпретировать как вероятность события, описываемого вектором $\mathbf{x}$ при условии события $\mathbf{x}_i$.

Основным отличием нормализованных радиальных базисных функций $\psi_N(\mathbf{x}, \mathbf{x}_i)$ (5.138) от обычных является знаменатель, являющийся *коэффициентом нормализации* (normalization factor). Этот коэффициент представляет собой оценку функции

плотности вероятности входного вектора $\mathbf{x}$. Следовательно, сумма базисных функций $\psi_N(\mathbf{x}, \mathbf{x}_i)$ по всем $i = 1, 2, \ldots, N$ дает в результате единицу (см. (5.139)). Выполнение этого условия для базисных функций (Грина) обычных сетей RBF (5.57) гарантировать нельзя.

При выводе формулы (5.138) для $F(\mathbf{x})$ основное внимание уделялось оценке плотности. Подобно задаче восстановления гиперповерхности, задача оценки плотности является плохо обусловленной. Функция оценки плотности Парзена–Розенблатта и, следовательно, функция оценки регрессии Надарайа–Ватсона могут быть выведены с применением теории регуляризации [1087]. Естественно, функционал стоимости для оценки плотности состоит из суммы двух слагаемых: среднеквадратической ошибки, включающей в себя неизвестную функцию плотности вероятности, и соответствующей формы функционала стабилизации.

Многомерное распределение Гаусса

В общем случае можно использовать множество разнообразных функций ядра. Однако теоретические и практические соображения ограничивают этот выбор. Как и в случае с функцией Грина, широко используемым ядром является многомерное распределение Гаусса:

$$K(\mathbf{x}) = \frac{1}{(2\pi)^{m_0/2}} \exp\left(-\frac{\|\mathbf{x}\|^2}{2}\right), \tag{5.142}$$

где m_0 — размерность входного вектора $\mathbf{x}$. Сферическая симметрия ядра $K(\mathbf{x})$ ясно прослеживается в формуле (5.142). Предполагая использование общей ширины (разброса) σ, играющей в распределении Гаусса роль параметра сглаживания h, и центрируя ядро в точке данных $\mathbf{x}_i$, можно записать, что

$$K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) = \frac{1}{(2\pi\sigma^2)^{m_0/2}} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}\right), \quad i = 1, 2, \ldots, N. \tag{5.143}$$

Исходя из этого, функция оценки регрессии Надарайа–Ватсона примет следующий вид [1011]:

$$F(\mathbf{x}) = \frac{\sum_{i=1}^{N} y_i \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}\right)}{\sum_{j=1}^{N} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_j\|^2}{2\sigma^2}\right)}, \tag{5.144}$$

где знаменатель представляет собой функцию оценки плотности Парзена–Розенблатта, состоящую из суммы N многомерных распределений Гаусса с центрами в точках данных $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.

Подставляя (5.143) в (5.138) и затем в (5.140), получим следующий вид функции отображения нормализованной сети RBF:

$$F(\mathbf{x}) = \frac{\sum_{i=1}^{N} w_i \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}\right)}{\sum_{j=1}^{N} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_j\|^2}{2\sigma^2}\right)}. \tag{5.145}$$

В выражениях (5.144) и (5.145) центры нормализованных радиальных базисных функций находятся в точках данных $\{\mathbf{x}_i\}_{i=1}^N$. Как и в случае с обычными радиальными базисными функциями, лучше использовать как можно меньшее число нормализованных RBF-функций, выбирая их центры, рассматриваемые как свободные параметры, согласно некоторой эвристике [750] или некоторому принципу [847].

5.13. Стратегии обучения

Процесс обучения сети на основе радиальных базисных функций (RBF) без учета его теоретического обоснования можно рассматривать следующим образом. Линейные веса, связанные с выходным узлом (узлами) сети, могут изменяться во "временном масштабе", отличном от используемого при работе нелинейных функций активации скрытых элементов. Если функции активации скрытых нейронов изменяются медленно, то веса выходных элементов изменяются довольно быстро с помощью линейной стратегии оптимизации. Здесь важно отметить, что разные слои сети RBF выполняют разные задачи, поэтому будет целесообразным отделить процесс оптимизации скрытого и выходного слоев друг от друга и использовать для них разные методы и, возможно, даже разные масштабы времени [676].

Существует множество различных стратегий обучения сети, зависящих от способа определения центров радиальных базисных функций. Первые три стратегии применимы к сетям RBF, описание которых основано на теории интерполяции. Последняя стратегия создания сетей сочетает в себе элементы теории регуляризации и оценки регрессии ядра.

Случайный выбор фиксированных центров

Простейший из подходов предполагает использование *фиксированных* радиальных базисных функций, определяющих функции активации скрытых элементов. Размещение центров может быть выбрано *случайным образом* из множества данных примеров. Такой подход считается "чувствительным" и требует *представительного* (representative) распределения множества обучающих данных с учетом рассматриваемой задачи [675]. Что же касается самих радиальных базисных функций, то для их реализа-

ции можно задействовать *изотропные функции Гаусса* (isotropic Gaussian function), стандартное отклонение которых является фиксированным относительно разброса центров. В частности, (нормализованные) радиальные базисные функции с центром в точке $\mathbf{t}_i$ определяются выражением

$$G(\|\mathbf{x}-\mathbf{t}_i\|^2)=\exp\left(\frac{m_1}{d_{\max}^2}\|\mathbf{x}-\mathbf{t}_i\|^2\right),\quad i=1,2,\ldots,m_1, \tag{5.146}$$

где m_1 — количество центров; d_{max} — максимальное расстояние между выбранными центрами. В результате стандартное отклонение (т.е. ширина) всех радиальных базисных функций Гаусса будет фиксированным:

$$\sigma=\frac{d_{\max}}{\sqrt{2m_1}}. \tag{5.147}$$

Эта формула гарантирует, что отдельные радиальные базисные функции не будут слишком гладкими или слишком остроконечными. Обоих этих крайних случаев следует избегать. В качестве альтернативы выражению (5.147) в наиболее разреженных областях можно использовать центры с большей шириной, что требует эксперимент с данными обучения.

При использовании этого подхода единственными параметрами, которые настраиваются в процессе обучения сети, являются синаптические веса ее выходного слоя. Их проще всего настроить с помощью *метода псевдообращения* (pseudoinverse method) [160]. В частности (см. (5.77) и (5.78)):

$$\mathbf{w}=\mathbf{G}^{+}\mathbf{d}, \tag{5.148}$$

где $\mathbf{d}$ — вектор желаемого отклика для множества примеров. Матрица $\mathbf{G}^{+}$ является псевдообратной матрице $\mathbf{G}$, которая, в свою очередь, определяется следующим образом:

$$\mathbf{G}=\{g_{ji}\}, \tag{5.149}$$

где

$$g_{ji}=\exp\left(-\frac{m_1}{d^2}\|\mathbf{x}_j-\mathbf{t}_i\|^2\right),\ j=1,2,\ldots,N;\ \ i=1,2,\ldots,m_1, \tag{5.150}$$

где $\mathbf{x}_j$ — j-й входной вектор множества примеров обучения.

Основой всех алгоритмов вычисления псевдообратных матриц является *сингулярная декомпозиция* (singular-value decomposition — SVD) [368].

Если $\mathbf{G}$ *— действительная матрица размерности* $N \times M$*, то существуют ортогональные матрицы*

$$\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N\}$$

и

$$\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M\},$$

такие, что

$$\mathbf{U}^T\mathbf{G}\mathbf{V} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_K), K = \min(N, M), \tag{5.151}$$

где

$$\sigma_1 \geq \sigma_2 \geq \quad \ldots \geq \sigma_K > 0.$$

Векторы-столбцы матрицы $\mathbf{U}$ называют *левыми сингулярными векторами* (left singular vector) матрицы $\mathbf{G}$, а векторы-столбцы матрицы $\mathbf{V}$ — *правыми сингулярными векторами* (left singular vector). Числа $\sigma_1, \sigma_2, \ldots, \sigma_K$ называют *сингулярными значениями* (singular value) матрицы $\mathbf{G}$. Согласно теореме о декомпозиции по сингулярным значениям, матрица, псевдообратная матрице $\mathbf{G}$, размерности $M \times N$ определяется следующим образом:

$$\mathbf{G}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T, \tag{5.152}$$

где $\mathbf{\Sigma}^+$ — матрица размерности $N \times N$, выраженная в терминах сингулярных значений матрицы $\mathbf{G}$:

$$\mathbf{\Sigma}^+ = \mathrm{diag}\left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, .., \frac{1}{\sigma_K}, 0, \ldots, 0\right). \tag{5.153}$$

Эффективный алгоритм вычисления псевдообратной матрицы описывается в [368].

Опыт использования методики случайного выбора центров показал, что этот метод относительно нечувствителен к использованию регуляризации. В задаче 5.14 будет предложено провести компьютерное моделирование задачи классификации с использованием этого метода. Случайный выбор центров можно использовать в качестве метода построения сетей RBF на основе множества примеров большого объема с возможным применением регуляризации.

Выбор центров на основе самоорганизации

Основной проблемой описанного выше метода выбора фиксированных центров является тот факт, что для обеспечения удовлетворительного уровня эффективности он требует большого множества примеров. Одним из способов обхода этой проблемы является использование *гибридного процесса обучения* (hybrid learning process), состоящего из двух этапов [185], [659], [750].

- *Этап обучения на основе самоорганизации* (self-organized learning stage). Его целью является оценка подходящих положений центров радиальных базисных функций скрытого слоя.
- *Этап обучения с учителем* (supervised learning stage). На этом этапе создание сети завершается оценкой линейных весов выходного слоя.

Хотя для реализации этих двух этапов обучения можно применить пакетную обработку, все-таки лучше использовать адаптивный (итеративный) подход.

Для процесса обучения на основе самоорганизации требуется разработать алгоритм *кластеризации*, разбивающий заданное множество точек данных на две подгруппы, каждая из которых должна быть максимально однородной. Один из таких алгоритмов называется *алгоритм кластеризации nok* — (k-means clasterization algorithm) [269]. Согласно этому алгоритму центры радиальных базисных функций размещаются только в тех областях входного пространства $\mathbf{X}$, в которых имеются информативные данные. Пусть m_1 — количество радиальных базисных функций. Определение подходящего значения для m_1 требует проведения некоторых экспериментов. Пусть $\{\mathbf{t}_k(n)\}_{k=1}^{m_1}$ — центры радиальных базисных функций на n-й итерации алгоритма. Тогда алгоритм кластеризации по k-средним можно описать следующим образом.

1. *Инициализация* (initialization). Выбираем случайные значения для исходных центров $\mathbf{t}_k(0)$. Единственным требованием к их выбору на данном шаге является различие всех начальных значений. При этом значения Евклидовой нормы по возможности должны быть небольшими.
2. *Выборка* (sampling). Выбираем вектор $\mathbf{x}$ из входного пространства $\mathbf{X}$ с определенной вероятностью. Этот вектор будет входным для алгоритма на итерации n.
3. *Проверка подобия* (similarity matching). Обозначим $k(\mathbf{x})$ индекс наиболее подходящего (победившего) центра для данного вектора $\mathbf{x}$. Находим $k(\mathbf{x})$ на итерации n, используя критерий минимального Евклидова расстояния:

$$k(\mathbf{x}) = \arg\min_k \|\mathbf{x}(n) - \mathbf{t}_k(n)\|, \quad k = 1, 2, \ldots, m_1, \tag{5.154}$$

где $\mathbf{t}_k(n)$ — центр k-й радиально базисной функции на итерации n.

4. *Корректировка* (updating). Корректируем центры радиальных базисных функций, используя следующее правило:

$$\mathbf{t}_k(n+1) = \begin{cases} \mathbf{t}_k(n) + \eta[\mathbf{x}(n) - \mathbf{t}_k(n)], & k = k(\mathbf{x}), \\ \mathbf{t}_k(n) \ - \text{ в противном случае}, \end{cases} \tag{5.155}$$

где η — *параметр скорости обучения* (learning-rate parameter), выбранный из диапазона $0 < \eta < 1$.

5. *Продолжение* (continuation). Увеличиваем на единицу значение n и возвращаемся к шагу 2, продолжая процедуру до тех пор, пока положение центров $\mathbf{t}_k$ существенно изменяется.

Описанный алгоритм кластеризации по k-средним на самом деле является *конкурентным* (competitive) процессом обучения, известным также под названием построения *карты самоорганизации* (self-organizing map). Более подробно он рассматривается в главе 9. Этот алгоритм целесообразно реализовывать на стадии обучения без учителя (на основе самоорганизации).

Ограничением алгоритма кластеризации по k-средним является нахождение только локального оптимального решения, зависящего от исходного выбора центров кластера. Следовательно, вычислительные ресурсы могут расходоваться напрасно: отдельные центры изначально попадут в те области входного пространства $\mathbf{X}$, где количество точек данных невелико и откуда не будет шанса переместиться в области, требующие большего количества центров. В результате можно получить неоправданно большую сеть. Чтобы обойти этот недостаток обычного алгоритма кластеризации по k-средним, в 1995 году был предложен *улучшенный алгоритм кластеризации по k-средним* (enhanced k-means clustering algorithm) [191], который основан на понятии *взвешенной переменной меры принадлежности кластеру* (cluster variation-weighted measure), обеспечивающем сходимость алгоритма к оптимальной или квазиоптимальной конфигурации, независимо от исходного положения центров.

Определив отдельные центры гауссовых радиальных базисных функций и их общий вес с помощью алгоритма кластеризации по k-средним или его улучшенной версии, можно перейти к следующему (и последнему) этапу процесса гибридного обучения — оценке весов выходного слоя. Простейшим методом такой оценки является алгоритм LMS (least-mean-square), описанный в главе 3. Вектор выходного сигнала, сгенерированного скрытыми узлами, является входным вектором алгоритма LMS. Обратите внимание на то, что алгоритм кластеризации по k-средним для скрытых узлов и алгоритм LMS для выходных узлов могут выполнять вычисления параллельно. Таким образом, процесс обучения ускоряется.

Выбор центров с учителем

В рамках третьего подхода центры радиальных базисных функций и все остальные свободные параметры сети настраиваются в процессе обучения с учителем. Другими словами, сеть RBF принимает самый общий вид. Естественным выбором для такой ситуации является обучение на основе коррекции ошибок, которое удобнее всего реализовывать с помощью процедуры градиентного спуска, являющейся обобщением алгоритма LMS.

Первым шагом в разработке такой процедуры обучения является определение значения функции стоимости:

$$\mathbf{E} = \frac{1}{2}\sum_{j=1}^{N} e_j^2, \tag{5.156}$$

где N — размер выборки, использованной для обучения; e_j — сигнал ошибки следующего вида:

$$e_j = d_j - F^*(\mathbf{x}_j) = d_j - \sum_{i=1}^{M} w_i G(\|\mathbf{x}_j - \mathbf{t}_i\|_{C_i}), \tag{5.157}$$

Требуется найти свободные параметры w_i, $\mathbf{t}_i$ и $\boldsymbol{\sigma}_i^{-1}$ (последний связан с матрицей взвешивания нормы $\mathbf{C}_i$), минимизирующие функцию стоимости $\mathbf{E}$. Результаты минимизации приведены в табл. 5.4. Следствия этих результатов представлены в упражнении 5.13. При рассмотрении табл. 5.4 особого внимания заслуживают следующие моменты.

- Функция стоимости $\mathbf{E}$ является выпуклой по линейному параметру w_i, однако не выпуклой по отношению к центрам $\mathbf{t}_i$ и матрице $\boldsymbol{\sigma}_i^{-1}$. В последних случаях поиск оптимального значения $\mathbf{t}_i$ и матрицы $\boldsymbol{\sigma}_i^{-1}$ может остановиться в точке локального минимума пространства параметров.
- В формулах для модификации значений w_i, $\mathbf{t}_i$ и $\boldsymbol{\sigma}_i^{-1}$ в общем случае можно использовать разные параметры скорости обучения η: η_1, η_2 и η_3.
- В отличие от алгоритма обратного распространения, процедура градиентного спуска (см. табл. 5.4) для сети RBF не предполагает обратного распространения сигнала ошибки.
- Вектор градиента $\partial\mathbf{E}/\partial\mathbf{t}_i$ обладает свойством, аналогичным свойству *алгоритма кластеризации* (clustering effect), — его значение зависит от *конкретной задачи* (task-dependent) [847].

ТАБЛИЦА 5.4. Формулы настройки линейных весов, положений и распределения центров для сетей RBF

1	Линейные веса (выходной слой) $\frac{\partial \mathbf{E}(n)}{\partial w_i(n)} = \sum_{j=1}^{N} e_j(n) G(\|\mathbf{x}_j - \mathbf{t}_i(n)\|_{C_i})$, $w_i(n+1) = w_i(n) - \eta_1 \frac{\partial \mathbf{E}(n)}{\partial w_i(n)}$, $i = 1, 2, \ldots, m_1$
2	Позиции центров (скрытый слой) $\frac{\partial \mathbf{E}(n)}{\partial \mathbf{t}_i(n)} = 2w_i(n) \sum_{j=1}^{N} e_j(n) G'(\|\mathbf{x}_j - \mathbf{t}_i(n)\|_{C_i}) \boldsymbol{\sigma}_i^{-1} [\mathbf{x}_j - \mathbf{t}_i(n)]$, $\mathbf{t}_i(n+1) = \mathbf{t}_i(n) - \eta_2 \frac{\partial \mathbf{E}(n)}{\partial \mathbf{t}_i(n)}$, $i = 1, 2, \ldots, m_1$
3	Распределение центров (скрытый слой) $\frac{\partial \mathbf{E}(n)}{\partial \boldsymbol{\sigma}_i^{-1}(n)} = -w_i(n) \sum_{j=1}^{N} e_j(n) G'(\|\mathbf{x}_j - \mathbf{t}_i(n)\|_{C_i}) \boldsymbol{\sigma}_i^{-1} \mathbf{Q}_{ji}(n)$, $\mathbf{Q}_{ji}(n) = [\mathbf{x}_j - \mathbf{t}_i(n)][\mathbf{x}_j - \mathbf{t}_i(n)]^T$, $\boldsymbol{\sigma}_i^{-1}(n+1) = \boldsymbol{\sigma}_i^{-1}(n) - \eta_3 \frac{\partial \mathbf{E}(n)}{\partial \boldsymbol{\sigma}_i^{-1}(n)}$,

где $e_j(n)$ — сигнал ошибки выходного узла j в момент времени n; $G'(\cdot)$ — первая производная функции Грина $G(\cdot)$ по своему аргументу

Для инициализации процедуры градиентного спуска поиск в пространстве параметров желательно начинать с некоторого *структурированного* начального условия, которое ограничивает область поиска уже известной полезной областью. Этого можно достичь с помощью стандартного метода классификации [676]. Таким образом, вероятность сходимости к нежелательному локальному минимуму в пространстве весов уменьшается. Например, можно начать с *гауссова классификатора* (Gaussian classifier), использование которого предполагает, что примеры всех классов имеют распределение Гаусса. Частный случай классификатора, основанного на процедуре проверки гипотез Байеса, см. в главе 3.

Возникает вопрос: чего можно добиться с помощью настройки положения центров радиальных базисных функций? Ответ на этот вопрос зависит от конкретной задачи. Тем не менее на основе опубликованных в литературе результатов можно сделать вывод, что идея настройки местоположения центров имеет некоторые преимущества. В работе, посвященной распознаванию речи с помощью сетей RBF [675], указано, что для нелинейной оптимизации параметров, определяющих функции активации скрытого слоя, желательно иметь сеть минимальной конфигурации. Однако, согласно [675], такой же производительности обобщения можно добиться, используя и большие сети RBF, т.е. сети с большим количеством фиксированных центров и настройкой выходного слоя с помощью линейной оптимизации.

В [1129] производительность сетей на основе (гауссовых) радиальных базисных функций с фиксированными центрами сравнивается с производительностью обоб-

щенных сетей на основе радиальных базисных функций с регулируемыми центрами (в последнем случае позиции центров определяются с помощью обучения с учителем). Сравнение выполнялось для задачи NETtalk. Первый эксперимент в этой области описан в [962], где использовался многослойный персептрон, обучаемый с помощью алгоритма обратного распространения. Более подробно он описывается в главе 13. Целью эксперимента, поставленного в [1129], являлся анализ обучения нейронных сетей фонетическому произношению английского текста. Из этого экспериментального исследования можно сделать следующие выводы.

- Сети RBF с самонастройкой (без учителя) положения центров и адаптацией (с учителем) весов выходного слоя не обеспечивают такой эффективности обобщения, как многослойный персептрон, обучаемый с помощью алгоритма обратного распространения.
- Обобщенные сети RBF (в которых обучение с учителем применяется как для настройки положения центров скрытого слоя, так и для адаптации весов выходного слоя) могут достичь уровня производительности многослойного персептрона.

Строгая интерполяция с регуляризацией

Метод построения сетей RBF, объединяющий в себе элементы теории регуляризации (см. раздел 5.5) и теории оценки регрессии ядра (см. раздел 5.12), описан в [1172]. Этот метод подразумевает использование четырех составляющих.

1. Радиальные базисные функции G, рассматриваемые (иногда с масштабированием) как ядро состоятельной оценки регрессии Надарайа–Ватсона (NWRE)[15].
2. Диагональная матрица взвешивания нормы $\mathbf{\Sigma}^{-1}$, общая для всех центров:

$$\mathbf{\Sigma} = \mathrm{diag}(h_1, h_2, \ldots, h_{m_0}), \tag{5.158}$$

   где $h_1, h_2, \ldots, \quad h_{m_0}$ — ширина полос по отдельным измерениям NWRE с (масштабированным) ядром G; m_0 — размерность входного пространства. Например, можно принять $h_i = \alpha_i \sigma_i^2$, где σ_i^2 — дисперсия i-й входной переменной, вычисленная на множестве входных данных. Затем можно определить *параметры масштабирования* (scaling factor) $\alpha_1, \alpha_2, \ldots, \alpha_{m_0}$, используя подходящую процедуру перекрестной проверки (см. раздел 5.9).
3. Регуляризированная процедура строгой интерполяции, включающая обучение линейных весов согласно (5.54).

---

[15] Метод оценки регрессии Надарайя-Ватсона является предметом интенсивного изучения в литературе по статистике. В более широком контексте непараметрическая функциональная оценка занимает центральное место в [418] и [907].

4. Выбор параметра регуляризации λ и множителей $\alpha_1, \alpha_2, \ldots, \alpha_{m_0}$ с помощью некоторого *ассимптотически-оптимального метода* (например, метода перекрестной проверки, определяемого выражением (5.117) или метода обобщенной перекрестной поверки, описываемого выражением (5.121)). Выбранные параметры можно интерпретировать следующим образом.
   - Чем больше значение параметра λ, тем сильнее шум влияет на входные параметры.
   - Если радиальная базисная функция G имеет унимодальное (т.е. гауссово) ядро, то чем меньше значения α_i, тем более "чувствителен" выход сети к соответствующей размерности входа. И наоборот, чем больше значения α_i, тем менее "адекватно" соответствующая входная размерность отражает влияние изменения входа на общий выход сети. Таким образом, выбранные значения α_i можно использовать для ранжирования важности соответствующих входных переменных и, следовательно, для определения потенциальных кандидатов на удаление при снижении размерности в случае необходимости.

Обоснование этой процедуры детально изложено в [1172]. В контексте рассматриваемой задачи такой выбор можно мотивировать следующим образом. Несложно показать, что сети NRWE составляют специальный класс регуляризированных сетей RBF в том смысле, что любую сеть NWRE можно аппроксимировать правильно построенной последовательностью регуляризированных сетей RBF, для которой последовательность параметров регуляризации $\{\lambda_N\}$ может возрастать до бесконечности с увеличением размера N обучающего множества (при этом среднеквадратическая или абсолютная ошибка аппроксимации будет стремиться к нулю). С другой стороны, при стремлении N к бесконечности риск, определяемый выражением (5.99), при определенных условиях будет стремиться к (глобальной) среднеквадратической ошибке. Если для построения последовательности параметров регуляризации используется процедура выбора асимптотически оптимальных параметров, то (по определению) результирующая последовательность RBF-сетей должна иметь (асимптотически) минимальную среднеквадратическую ошибку на всем множестве возможных последовательностей параметров (включая ту, которая соответствует сети NWRE). Если известно, что сеть NWRE является состоятельной в смысле среднеквадратической ошибки, то этому же условию удовлетворяет регуляризированная сеть RBF, построенная согласно такой же процедуре. Другими словами, регуляризированная сеть RBF, построенная с помощью этой процедуры, может унаследовать свойства состоятельности сети NWRE. Такая взаимозависимость позволяет распространить известные результаты, касающиеся состоятельности сетей NWRE, на такие области, как *регрессия временных рядов* (time series regression), в которых часто встречаются *зависимые* и *нестационарные* процессы и где допущения обычных нейронных сетей о независимости данных обучения и стационарности процессов не выполняются. Подводя итог,

можно сказать, что описанная здесь процедура, объединяющая элементы теории регуляризации и теории оценки регрессии ядра, обеспечивает практическую методику создания и применения теоретически обоснованных регуляризированных сетей RBF.

5.14. Компьютерное моделирование: классификация образов

В этом разделе описывается компьютерный эксперимент, иллюстрирующий методику построения регуляризированной сети RBF на основе использования строгой интерполяции. В этом эксперименте решается задача бинарной классификации данных, полученных на основе двух равновероятных перекрывающихся двумерных распределений Гаусса, соответствующих классам $\mathbf{C}_1$ и $\mathbf{C}_2$. Параметры распределения Гаусса выбраны такими же, как и в разделе 4.8. Класс $\mathbf{C}_1$ характеризуется вектором среднего значения $[0,0]^T$ и дисперсией, равной единице; класс $\mathbf{C}_2$ характеризуется вектором среднего значения $[0,2]^T$ и дисперсией, равной 4. Пример, описываемый в этом разделе, можно рассматривать как аналог эксперимента, описанного в разделе 4.8, для обучения методом обратного распространения, реализованный для регуляризированной сети RBF.

Для двух классов, $\mathbf{C}_1$ и $\mathbf{C}_2$, строится регуляризированная сеть RBF с двумя выходными функциями — по одной для каждого класса. Кроме того, в качестве желаемых используются следующие бинарные значения:

$$d_k^{(p)} = \begin{cases} 1\ , & \text{если пример } p \text{ принадлежит классу } \mathbf{C}_k, \\ 0\ - & \text{в противном случае,} \end{cases}$$

где $k = 1, 2$.

Прежде чем приступить к эксперименту, следует определить решающее правило, которое будет применяться для классификации. В [1172] показано, что выходы регуляризированной сети RBF обеспечивают оценку апостериорной вероятности принадлежности классу. Однако это истинно только при условии, что сеть обучается с помощью двоичных векторных индикаторов желаемого отклика класса. Тогда для этого класса сетей можно применить следующее решающее правило (4.55).

Выбирается класс, соответствующий максимальному выходному значению функции.

Метод строгой интерполяции для выбора центров проверялся для различных значений параметра регуляризации λ. При заданном λ для вычисления вектора весов выходного слоя сети RBF использовалось выражение (5.54):

$$\mathbf{w} = (\mathbf{G} + \lambda \mathbf{I})^{-1}\mathbf{d},$$

ТАБЛИЦА 5.5. Вероятность корректной классификации $\mathbf{P}_c$(%) для разных значений параметра регуляризации λ и размера скрытого слоя $m_1 = 20$

	Значение параметра регуляризации λ					
Вид статистики	0	0,1	1	10	100	1000
Среднее	57,49	72,42	74,42	73,80	72,46	72,14
Стандартное отклонение	7,47	4,11	3,51	4,17	4,98	5,09
Минимум	44,20	61,60	65,80	63,10	60,90	60,50
Максимум	72,70	78,30	78,90	79,20	79,40	79,40

ТАБЛИЦА 5.6. Вероятность корректной классификации $\mathbf{P}_c$(%) для разных значений параметра регуляризации λ и размера скрытого слоя $m_1 = 100$

	Значение параметра регуляризации λ					
Вид статистики	0	0,1	1	10	100	1000
Среднее	50,58	77,03	77,72	77,87	76,47	75,33
Стандартное отклонение	4,70	1,45	0,94	0,91	1,62	2,25
Минимум	41,00	70,60	75,10	75,10	72,10	70,10
Максимум	61,30	79,20	79,80	79,40	78,70	78,20

где $\mathbf{G}$ — матрица Грина размерности $N \times N$, ji-й элемент которой соответствует значению радиальной симметричной функции Грина $G(\mathbf{x}_j, \mathbf{x}_i)$; N — размер обучающего множества; $\mathbf{d}$ — вектор желаемого отклика.

Для каждого из параметров регуляризации строились 50 независимых сетей, которые тестировались на одном и том же множестве из 1000 примеров.

В табл. 5.5 приведена обобщенная статистическая информация о вероятности корректной классификации P_c, вычисленной для случая m_1 =20 центров. Статистика по массиву вычислялась для различных значений параметра регуляризации λ. В табл. 5.6 представлены соответствующие результаты, вычисленные для случая регуляризированной сети RBF большего размера с $m_1 = 100$ центрами.

На рис. 5.7 показаны границы решений, сформированные выходами сети для значения параметра регуляризации $\lambda = 10$ (при таком значении параметра статистические характеристики оказались наилучшими). На рис. 5.7, *а* показана сеть с наилучшими показателями, а на рис. 5.7, *б* — с наихудшими статистическими показателями в рамках массива. Обе части этого рисунка представлены для случая сети с $m_1 = 100$ центрами.

Сравнивая табл. 5.5 и 5.6, можно сделать следующие выводы.

1. Как в случае m_1 =20 центров, так и в случае $m_1 = 100$ центров качество классификации в сети для параметра регуляризации λ =0 остается относительно плохим.

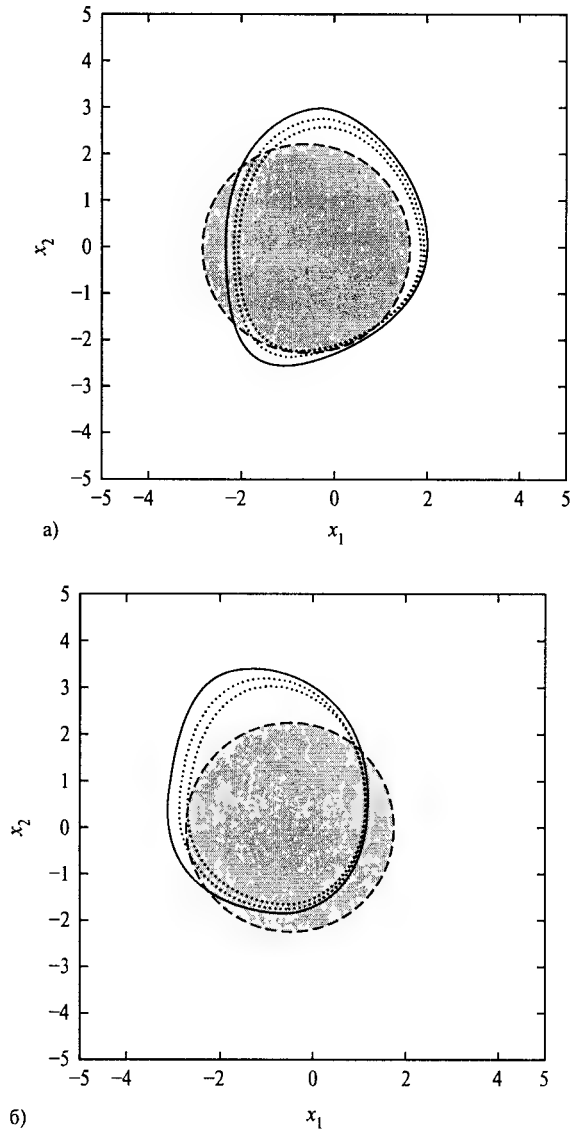


Рис. 5.7. Результаты компьютерного моделирования при решении задачи классификации на основе строгой интерполяции в регуляризируемых сетях RBF: наилучшее решение (а); наихудшее решение (б). Закрашенная область представляет собой оптимальное решение Байеса

2. Использование регуляризации оказывает большое влияние на качество классификации в сетях RBF.
3. Для $\lambda \geq 0,1$ качество классификации зависит от значения λ. Для случая m_1 =20 центров лучшие показатели были достигнуты при $\lambda = 1$, а для случая $m_1 = 100$ центров — при $\lambda = 10$.
4. Увеличение количества центров с 20 до 100 улучшает качество классификации приблизительно на 4,5%.

5.15. Резюме и обсуждение

Структура сетей RBF является необычной в том смысле, что архитектура скрытых элементов в корне отличается от структуры выходных. Поскольку основой функционирования нейронов скрытого слоя являются радиальные базисные функции, теория сетей RBF тесно связана с теорией радиальных базисных функций, которая в настоящее время является одной из основных областей изучения в численном анализе [995]. Интересным также является тот факт, что настройка линейных весов выходного слоя позволяет обеспечить хорошее качество классификации. В этом можно удостовериться, изучив литературу по адаптивным линейным фильтрам [434], [435].

В отличие от многослойных персептронов, обучаемых алгоритмом обратного распространения, архитектура сетей RBF создается в соответствии с некоторыми принципиальными установками. В частности, теория регуляризации Тихонова, вкратце рассмотренная в разделе 5.5, является прочным математическим фундаментом для теории сетей RBF. В этой теории главную роль играет функция Грина $G(\mathbf{x}, \boldsymbol{\xi})$. Форма функции Грина как фундаментальной функции сети определяется формой *ограничения на гладкость* (smoothing constraint), задаваемого в теории регуляризации. Это условие, определяемое оператором дифференцирования $\mathbf{D}$ из (5.63), приводит к построению многомерной функции Гаусса для функции Грина. Используя различные варианты оператора дифференцирования $\mathbf{D}$, можно прийти к совершенно различным формам функции Грина. Напомним, что при ослаблении требования на меньшее количество базисных функций по сравнению с количеством точек важным фактором в определении сглаживающего регуляризатора становится уменьшение вычислительной сложности. Это является еще одной причиной использования некоторых других функций в качестве базисных для регуляризируемых сетей RBF (см. рис. 5.5). Каким бы ни был выбор базисных функций, для получения наилучшего результата применения теории регуляризации к сетям RBF требуется принципиальный подход к оценке параметра регуляризации λ. Этому требованию удовлетворяет обобщенная перекрестная проверка, описанная в разделе 5.9. Теория, обосновывающая применение обобщенной перекрестной проверки, является асимптотической. В ней предусмотрено следующее условие: для достижимости хорошей оценки параметра регуляризации λ множество примеров должно быть достаточно большим.

Еще одним принципиальным подходом к построению сетей RBF является теория регрессии ядра. В этом подходе на вооружение берется оценка плотности, для которой радиально базисные функции в сумме дают единицу. Многомерное распределение Гаусса обеспечивает удобный метод удовлетворить это требование.

В заключение отметим, что функции отображения в гауссовых сетях RBF напоминают функции, реализованные на основе учета мнения группы экспертов, которые рассматриваются в главе 7.

Задачи

Радиальные базисные функции

5.1. Рассмотрим функцию следующего вида:

$$\varphi(r) = \left(\frac{r}{\sigma}\right)^2 \log\left(\frac{r}{\sigma}\right), \quad \text{где } \sigma > 0 \text{ и } r \in \Re.$$

Обоснуйте использование этой функции в качестве инварианта для операций вращения и переноса для функции Грина.

5.2. Множество значений, представленных в разделе 5.8 для вектора весов **w** в сети RBF, показанной на рис. 5.6, представляет одно из возможных решений задачи XOR. Исследуйте еще одно множество значений вектора весов **w** для решения той же задачи.

5.3. В разделе 5.8 представлено решение задачи XOR с помощью сети RBF, имеющей два скрытых элемента. В этой задаче рассмотрим точное решение задачи XOR с помощью сети RBF с четырьмя скрытыми элементами. Центры радиальных базисных функций определяются соответствующими образами входных данных. Четыре возможных входных примера определены координатами (0,0), (0,1), (1,0) и (1,1), представляющими циклический обход вершин квадрата.

а) Постройте матрицу интерполяции **Φ** для полученной сети RBF и вычислите для нее обратную.

б) Вычислите линейные веса выходного слоя этой сети.

5.4. Функция Гаусса является единственной факторизуемой радиальной базисной функцией.

Используя это свойство, покажите, что функция Грина $G(\mathbf{x},\mathbf{t})$, определенная как многомерное распределение Гаусса, может быть факторизована следующим образом:

$$G(\mathbf{x}, \mathbf{t}) = \prod_{i=1}^{m} G(x_i, t_i),$$

где x_i и t_i являются соответственно i-ми элементами векторов **x** и **t**.

Сети регуляризации

5.5. Рассмотрим следующий функционал стоимости:

$$\mathbf{E}(F^*) = \sum_{i=1}^{N} \left[d_i - \sum_{j=1}^{m_1} w_j G\left(\|\mathbf{x}_j - \mathbf{t}_i\|\right) \right]^2 + \lambda \|\mathbf{D}F^*\|^2,$$

который относится к функции аппроксимации

$$F^*(\mathbf{x}) = \sum_{j=1}^{m_1} w_j G\left(\|\mathbf{x} - \mathbf{t}_i\|\right).$$

Используя дифференциал Фреше, покажите, что функционал стоимости $\mathbf{E}(F^*)$ достигает минимума при

$$(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{G}_0)\mathbf{w} = \mathbf{G}^T\mathbf{d},$$

где $\mathbf{G}$ — матрица размерности $N \times m_1$; $\mathbf{G}_0$ — матрица размерности $m_1 \times m_1$; $\mathbf{w}$ — вектор размерности $m_1 \times 1$; $\mathbf{d}$ — вектор размерности $N \times 1$. Эти элементы определяются соответственно выражениями (5.72), (5.73), (5.75) и (5.46).

5.6. Предположим, что определено соотношение

$$(\tilde{\mathbf{D}}\mathbf{D})_{\mathbf{U}} = \sum_{k=0}^{\infty} (-1)^k \frac{\nabla_{\mathbf{U}}^{2k}}{k!2^k},$$

где

$$\nabla_{\mathbf{U}}^2 = \sum_{j=1}^{m_0} \sum_{i=1}^{m_0} u_{ji} \frac{\partial^2}{\partial x_j \partial x_i}.$$

Матрица $\mathbf{U}$ размерности $m_0 \times m_0$, ji-й элемент которой обозначается как u_{ji}, является симметрической и положительно определенной. Исходя из этого, для нее существует обратная матрица $\mathbf{U}^{-1}$. Это позволяет выполнить следующую декомпозицию:

$$\mathbf{U}^{-1} = \mathbf{V}^T\sigma\mathbf{V} = \mathbf{V}^T\sigma^{1/2}\sigma^{1/2}\mathbf{V} = \mathbf{C}^T\mathbf{C},$$

где $\mathbf{V}$ — ортогональная матрица; $\boldsymbol{\sigma}$ — диагональная матрица; $\boldsymbol{\sigma}^{1/2}$ — квадратный корень из последней, а матрица $\mathbf{C}$ определяется следующим образом:

$$\mathbf{C} = \boldsymbol{\sigma}^{1/2}\mathbf{V}.$$

Требуется решить задачу нахождения функции Грина $G(\mathbf{x}, \mathbf{t})$, удовлетворяющей следующему условию (в смысле распределения):

$$(\tilde{\mathbf{D}}\mathbf{D})_{\mathrm{U}} G(\mathbf{x}, \mathbf{t}) = \delta(\mathbf{x} - \mathbf{t}).$$

Используя многомерное преобразование Фурье для решения этого уравнения для $G(\mathbf{x}, \mathbf{t})$, покажите, что

$$G(\mathbf{x}, \mathbf{t}) = \exp\left(-\frac{1}{2}\left\|\mathbf{x} - \mathbf{t}\right\|_C^2\right),$$

где

$$\left\|\mathbf{x}\right\|_C^2 = \mathbf{x}^T\mathbf{C}^T\mathbf{C}\mathbf{x}.$$

5.7. Рассмотрим слагаемое регуляризации, представленное в следующем виде:

$$\int_{\Re^{m_0}} \left\|\mathbf{D}F(\mathbf{x})\right\|^2 d\mathbf{x} = \sum_{k=0}^{\infty} a_k \int_{\Re^{m_0}} \left\|D^k F(\mathbf{x})\right\|^2 d\mathbf{x},$$

где

$$a_k = \frac{\sigma^{2k}}{k!2^k}.$$

Пусть линейный оператор дифференцирования определен в терминах оператора градиента ∇ и оператора Лапласиана ∇^2 следующим образом:

$$D^{2k} = (\nabla^2)^k$$

и

$$D^{2k+1} = \nabla(\nabla^2)^k.$$

Покажите, что

$$\mathbf{D}F(\mathbf{x}) = \sum_{k=0}^{\infty} \frac{\sigma^{2k}}{k!2^k} \nabla^{2k} f(\mathbf{x}).$$

5.8. В разделе 5.5 с помощью соотношения (5.65) была выведена функция аппроксимации $F_\lambda(\mathbf{x})$, представленная в выражении (5.66). В данной задаче для получения формулы (5.66) воспользуемся выражением (5.65) и многомерным преобразованием Фурье. Выведите эту формулу, используя следующее определение многомерного преобразования Фурье функции Грина $G(\mathbf{x})$:

$$G(\mathbf{s}) = \int_{\Re^{m_0}} G(\mathbf{x}) \exp(-i\,\mathbf{s}^T\mathbf{x}) d\mathbf{x},$$

где $i = \sqrt{-1}$; $\mathbf{s}$ — m_0-мерная переменная преобразования.

5.9. Рассмотрим задачу нелинейной регрессии, описываемую выражением (5.95). Пусть a_{ik} — ik-й элемент обратной матрицы $(\mathbf{G} + \lambda\mathbf{I})^{-1}$. Исходя из этого, начиная с формулы (5.58), покажите, что оценка функции регрессии $f(\mathbf{x})$ может быть описана как

$$\hat{f}(\mathbf{x}) = \sum_{k=1}^{N} \psi(\mathbf{x}, \mathbf{x}_k) y_k,$$

где y_k — выход модели, соответствующий входному сигналу $\mathbf{x}_k$, и

$$\psi(\mathbf{x}, \mathbf{x}_k) \sum_{i=1}^{N} G(\|\mathbf{x} - \mathbf{x}_i\|) a_{ik}, k = 1,\ 2, \ldots, N,$$

где $G(\|\cdot\|)$ — функция Грина.

5.10. *Сплайн-функция* (spline function) представляет собой пример полиномиального аппроксиматора [950]. Основная идея, положенная в основу этого метода, заключается в следующем. Область аппроксимации разбивается на конечное число частей посредством *деления* (knot), которое может быть фиксированным. В этом случае аппроксимация является *линейно*-параметризованной. Однако если деление не равномерное, аппроксимация считается *нелинейно*-параметризованной. В обоих случаях на каждом из участков для аппроксимации используется полином степени не меньшей n. Единственным условием является $(n-1)$-кратная дифференцируемость всей функции. Полиномиальные сплайны являются относительно гладкими функциями, которые легко хранить, выполнять с ними операции и оценивать на компьютере.

Среди сплайн-функций, реально используемых на практике, самыми популярными являются *кубические сплайны* (cubic spline). Функция стоимости для такого сплайна (подразумевая одномерный входной сигнал) определяется следующим выражением:

$$\mathbf{E}(f) = \frac{1}{2}\sum_{i=1}^{N}[y_i - f(x_i)]^2 + \frac{\lambda}{2}\int_{x_1}^{x_N}\left[\frac{d^2 f(x)}{dx^2}\right]dx,$$

где на языке сплайнов λ называется *параметром сглаживания* (smoothing parameter).

а) Обоснуйте следующие свойства решения $f_\lambda(x)$.
   1. Функция $f_\lambda(x)$ является кубическим полиномом между двумя точными значениями x.
   2. Функция $f_\lambda(x)$ и ее первые две производные являются непрерывными, за исключением граничных точек, в которых значение второй производной равно нулю.

б) Так как функция стоимости $\mathbf{E}(f)$ имеет единственный минимум, для любого g, взятого из того же класса дважды дифференцируемых функций, что и f_λ, выполняется соотношение

$$\mathbf{E}(f_\lambda + \alpha g) \geq \mathbf{E}(f_\lambda),$$

где α — действительная положительная константа. Это значит, что $\mathbf{E}(f_\lambda + \alpha g)$, интерпретируемая как функция от α, должна иметь локальный минимум в точке $\alpha = 0$. Исходя из этого, покажите, что

$$\int_{x_1}^{x_N}\left(\frac{d^2 f_\lambda(x)}{dx^2}\right)\left(\frac{d^2 g(x)}{dx^2}\right)dx = \frac{1}{2}\sum_{i=1}^{N}[y - f_\lambda(x_i)]g(x_i).$$

Это соотношение называется уравнением Эйлера–Лагранжа для задачи кубического сплайна.

Порядок аппроксимации

5.11. Уравнение (5.124) определяет верхнюю границу ошибки обобщения в гауссовой сети RBF, созданной для обучения функции регрессии, принадлежащей определенному пространству Соболева. Используя это ограничение, выведите формулу (5.125) оптимального размера такой сети для заданного размера множества примеров обучения.

Оценка ядра

5.12. Предположим, что задано не содержащее шума обучающее множество $\{f(\mathbf{x}_i)\}_{i=1}^N$. Требуется построить сеть, которая обобщает данные, искаженные сторонним шумом и не включенные в набор примеров обучения. Пусть $F(\mathbf{x})$ — функция аппроксимации, реализуемая такой сетью и выбранная так, чтобы ожидаемая квадратичная ошибка

$$J(F) = \frac{1}{2}\sum_{i=1}^{N}\int_{\Re^{m_0}}[f(\mathbf{x}_i) - F(\mathbf{x}_i, \boldsymbol{\xi})]^2 f_{\xi}(\boldsymbol{\xi})d\boldsymbol{\xi}$$

достигала своего минимума. Здесь $f_{\xi}(\boldsymbol{\xi})$ — функция плотности вероятности распределения шума в пространстве входного сигнала $\Re^{m_0}$. Покажите, что решение этой задачи задается следующей формулой:

$$F(\mathbf{x}) = \frac{\sum_{i=1}^{N} f(\mathbf{x}_i) f_{\xi}(\mathbf{x} - \mathbf{x}_i)}{\sum_{i=1}^{N} f_{\xi}(\mathbf{x} - \mathbf{x}_i)}.$$

Сравните эту оценку с оценкой регрессии Надарайя–Ватсона.

Выбор центров с учителем

5.13. Рассмотрим функционал стоимости

$$\mathbf{E} = \frac{1}{2}\sum_{j=1}^{N} e_j^2,$$

где

$$e_j = d_j - F^*(x_j) = d_j - \sum_{i=1}^{m_1} w_i G(\|\mathbf{x}_j - \mathbf{t}_i\|_{C_i}).$$

Свободными параметрами являются линейные веса w_i, центры $\mathbf{t}_i$ функций Грина и обратная матрица ковариации $\boldsymbol{\sigma}_i^{-1} = \mathbf{C}_i^T\mathbf{C}_i$, где $\mathbf{C}_i$ — матрица взвешивания нормы. Задача заключается в нахождении свободных параметров, которые минимизируют функционал стоимости $\mathbf{E}$. При этом можно воспользоваться следующими частными производными:

а) $\frac{\partial \mathbf{E}}{\partial w_i} = \sum_{j=1}^{N} e_j G(\|\mathbf{x}_j - \mathbf{t}_i\|_{C_i})$,

б) $\frac{\partial \mathbf{E}}{\partial \mathbf{t}_i} = 2w_i \sum_{j=1}^{N} e_j G'(\|\mathbf{x}_j - \mathbf{t}_i\|_{C_i})\boldsymbol{\sigma}_i^{-1}(\mathbf{x}_j - \mathbf{t}_i)$,

в) $\frac{\partial \mathbf{E}}{\partial \boldsymbol{\sigma}_i^{-1}} = -w_i \sum_{j=1}^{N} e_j G'(\|\mathbf{x}_j - \mathbf{t}_i\|_{C_i}) \mathbf{Q}_{ji}$,

где $G'(\cdot)$ — производная $G(\cdot)$ по ее аргументу и

$$\mathbf{Q}_{ji} = (\mathbf{x}_j - \mathbf{t}_i)(\mathbf{x}_j - \mathbf{t}_i)^T.$$

Правила дифференцирования скаляра по вектору см. в примечании 2 к главе 3.

Компьютерное моделирование

5.14. В этом задании мы продолжим компьютерный эксперимент, начатый в разделе 5.13, с целью изучения возможностей выбора центров при создании сети RBF для двоичного классификатора. Целью настоящего эксперимента будет демонстрация того факта, что качество обобщения в сети такого типа будет довольно неплохим.

С помощью сети, описанной в разделе 5.13 и предназначенной для решения задачи двоичной классификации, решается задача классификации данных, выбранных из смешанной модели, состоящей из двух равновероятных пересекающихся гауссовых распределений. Одно из распределений имеет вектор среднего значения $[0, 1]^T$ и общую дисперсию, равную единице. Другое распределение имеет среднее значение $[0, 2]^T$ и общую дисперсию 4. Для классификации необходимо использовать решающее правило: "выбрать класс, дающий максимальное значение выходного сигнала".

а) Рассмотрите случайный выбор центров для $m_1 = 20$. Подсчитайте среднее значение, стандартное отклонение, максимальное и минимальное значения вероятности корректной классификации P_c для различных значений параметра регуляризации $\lambda = 0, 0.1, 1, 10, 100, 1000$. Для вычисления общей статистики используйте 50 независимых видов сетей, проверяя каждый вид на фиксированном множестве из 1000 примеров.

б) Постройте границу решений для конфигурации, описанной в предыдущем пункте для значения параметра регуляризации $\lambda = 1$.

в) Повторите вычисления пункта а) для $m_1 = 10$ центров (выбираемых случайно).

г) В свете полученных результатов опишите преимущества случайного выбора центров, применяемого в качестве метода построения сетей RBF. Оцените роль регуляризации в общей производительности сети, выступающей в качестве классификатора.

д) Сравните полученные результаты с описанными в разделе 5.13, где использовался метод строгой интерполяции. В частности, подтвердите, что случайный выбор центров относительно нечувствителен к параметру регуляризации.

5.15. Можно доказать, что для эксперимента, описанного в разделе 5.13 и проводимого для классификации двух классов с гауссовым распределением, сеть RBF достаточно хорошо зарекомендовала себя благодаря использованию гауссовых радиальных базисных функций для аппроксимации рассматриваемых условных распределений Гаусса. В настоящей задаче воспользуемся методом компьютерного моделирования для рассмотрения сети RBF с разрывными условными распределениями Гаусса. В частности, рассмотрим два класса, $\mathbf{C}_1$ и $\mathbf{C}_2$, со следующими распределениями.

- $U(\mathbf{C}_1)$, где $\mathbf{C}_1 \triangleq \Omega_1$ — окружность с радиусом $r = 2.34$ и центром в точке $\mathbf{x}_c = [-2, 30]^T$.
- $U(\mathbf{C}_2)$, где $\mathbf{C}_2 \subset \Re^2$ — квадрат с центром в точке $\mathbf{x}_c = [-2, 30]^T$ и длиной стороны $r = \sqrt{2\pi}$.

Здесь под $U(\mathbf{C}_1)$ понимается равномерное распределение на $\Omega \subset \Re^2$. Эти параметры выбираются таким образом, чтобы область решений для класса $\mathbf{C}_1$ совпадала со случаем распределения Гаусса, рассмотренного в разделе 5.13. Исследуйте применение регуляризации как средства повышения качества классификации в гауссовых сетях RBF при использовании строгой интерполяции.

Машины опорных векторов

6.1. Введение

В главе 4 рассматривались многослойные персептроны, обучаемые по алгоритму обратного распространения ошибки. В главе 5 исследовался другой класс многослойных сетей прямого распространения — сети на основе радиальных базисных функций. Оба эти типа нейронных сетей являются универсальными аппроксиматорами, каждый в своем смысле. В этой главе будет представлена еще одна категория универсальных сетей прямого распространения — так называемые *машины опорных векторов* (support vector machine — SVM), предложенные Вапником [141], [212], [1084], [1085]. Подобно многослойным персептронам и сетям на основе радиальных базисных функций, машины опорных векторов можно использовать для решения задач классификации и нелинейной регрессии.

Машина опорных векторов — это *линейная система* (linear machine), обладающая рядом привлекательных свойств. Описание работы таких машин следует начать с вопроса разделимости классов, возникающего при решении задач классификации. В этом контексте идея машин опорных векторов состоит в построении гиперплоскости, выступающей в качестве поверхности решений, максимально разделяющей положительные и отрицательные примеры. Это достигается благодаря принципиальному подходу, основанному на теории статистического обучения (см. главу 2). Более конкретно, машина опорных векторов является аппроксимирующей реализацией *метода минимизации структурного риска* (method of structural risk minimization). Этот индуктивный принцип основан на том, что уровень ошибок обучаемой машины на данных тестирования (т.е. уровень ошибок обобщения) можно представить в виде суммы ошибки обучения и слагаемого, зависящего от *измерения Вапника–Червоненкиса* (Vapnik-Chervonenkis dimension). В случае разделяемых множеств машина опорных векторов выдает значение "нуль" для первого слагаемого, минимизируя при этом второе слагаемое. Поэтому машина опорных векторов может обеспечить хорошее качество обобщения в задаче классификации, не обладая априорными знаниями о предметной области конкретной задачи. Именно это свойство является уникальным для машин опорных векторов.

Понятие, лежащее в основе построения алгоритма обучения опорных векторов, — это ядро скалярного произведения "опорного вектора" $\mathbf{x}_i$ и вектора $\mathbf{x}$, взятого из входного пространства. Опорные векторы представляют собой небольшое подмножество обучающих данных, отбираемых алгоритмом. В зависимости от метода генерации этого ядра можно построить различные обучаемые машины со своими собственными нелинейными поверхностями решений. В частности, алгоритм настройки опорных векторов можно использовать для построения следующих трех типов обучаемых машин (и не только их).

- Полиномиальные обучаемые машины.
- Сети на основе радиальных базисных функций.
- Двухслойные персептроны (т.е. с одним скрытым слоем).

Это значит, что для каждой из этих сетей прямого распространения можно реализовать процесс обучения на основе алгоритма настройки опорных векторов, использующего предложенный набор данных обучения для автоматического определения количества необходимых скрытых элементов. Другими словами, если алгоритм обратного распространения создан специально для обучения многослойных персептронов, то алгоритм обучения опорных векторов носит более общий характер, так как имеет более широкую область применения.

Структура главы

Данная глава состоит из трех основных частей. В первой части описываются основные идеи, положенные в основу машин опорных векторов. В частности, в разделе 6.2 описывается процесс построения оптимальных гиперплоскостей для простейшего случая линейно-разделимых множеств. В разделе 6.3 рассматривается более сложный случай неразделимых множеств.

Таким образом, мы подготовим почву для второй части настоящей главы, в которой детально описывается машина опорных векторов, предназначенная для решения задач классификации. Эти вопросы освещаются в разделе 6.4. В разделе 6.5 мы вернемся к задаче исключающего ИЛИ, на которой продемонстрируем процесс создания машины опорных векторов. Раздел 6.6 посвящен компьютерному эксперименту по решению задачи классификации, рассмотренной в главах 4 и 5. Таким образом, мы сможем произвести сравнительную оценку машин опорных векторов с многослойными персептронами, обучаемыми по методу обратного распространения, а также с сетями на основе радиальных базисных функций.

В последней части главы рассматривается задача нелинейной регрессии. В разделе 6.7 описывается функция потерь (loss function), которая лучше всего подходит для данной задачи. В разделе 6.8 обсуждается создание машины опорных векторов для решения задачи нелинейной регрессии.

В разделе 6.9 приводятся выводы и некоторые заключительные замечания.

6.2. Оптимальная гиперплоскость для линейно-разделимых образов

Рассмотрим множество обучающих примеров $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, где $\mathbf{x}_i$ — входной образ для примера i; d_i — соответствующий ему желаемый отклик (целевой выход). Для начала предположим, что класс, представленный подмножеством $d_i = +1$, и класс, представленный подмножеством $d_i = -1$, линейно-разделимы. Уравнение поверхности решений в форме гиперплоскости, выполняющей это разделение, записывается следующим образом:

$$\mathbf{w}^T\mathbf{x} + b = 0, \tag{6.1}$$

где $\mathbf{x}$ — входной вектор; $\mathbf{w}$ — настраиваемый вектор весов; b — порог. Таким образом, можно записать:

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b \geq 0 \text{ для } d_i = +1, \\ \mathbf{w}^T\mathbf{x}_i + b < 0 \text{ для } d_i = -1. \end{aligned} \tag{6.2}$$

Допущение о линейной разделимости образов введено для того, чтобы доступно объяснить основную идею, положенную в основу машин опорных векторов. Далее, в разделе 6.3, это допущение будет ослаблено.

Для данного вектора весов $\mathbf{w}$ и порога b расстояние между гиперплоскостью, задаваемой уравнением (6.1), и ближайшей точкой из набора данных называется *границей разделения* (margin of separation) и обозначается символом ρ. Основной целью машины опорных векторов является поиск конкретной гиперплоскости, для которой граница разделения будет максимальной. При этом условии поверхность решения называется *оптимальной гиперплоскостью* (optimal hyperplane). На рис. 6.1 показано геометрическое представление оптимальной гиперплоскости в двумерном пространстве входных сигналов.

Пусть $\mathbf{w}_o$ и b_o — оптимальные значения вектора весов и порога соответственно. Исходя из этого, *оптимальную гиперплоскость*, представляющую многомерную линейную поверхность решений в пространстве входных сигналов, можно описать уравнением

$$\mathbf{w}_o^T\mathbf{x} + b_o = 0. \tag{6.3}$$

Оно является аналогом уравнения (6.1). При этом дискриминантная функция

$$g(\mathbf{x}) = \mathbf{w}_o^T\mathbf{x} + b_o \tag{6.4}$$

определяет алгебраическую меру расстояния от точки $\mathbf{x}$ до оптимальной гиперплоскости [269].

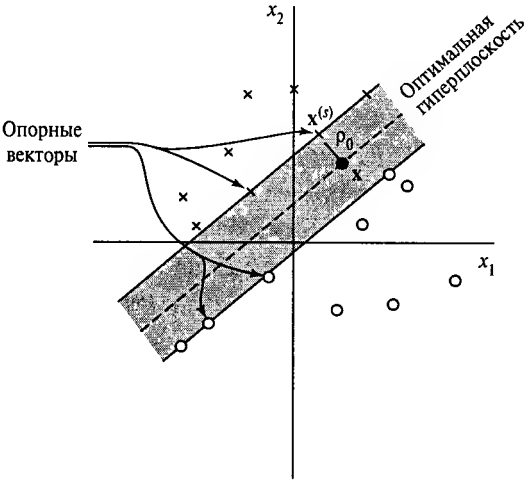


Рис. 6.1. Оптимальная гиперплоскость для линейно-разделимых образов

Это легко увидеть, если выразить $\mathbf{x}$ следующим образом:

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}_o}{\|\mathbf{w}_o\|},$$

где $\mathbf{x}_p$ — нормальная проекция точки $\mathbf{x}$ на оптимальную гиперплоскость; r — желаемое алгебраическое расстояние. Величина r является положительной, если $\boldsymbol{x}$ находится с положительной стороны оптимальной гиперплоскости, и отрицательным в противном случае. Так как по определению $g(\mathbf{x}_p) = 0$, следовательно,

$$g(\mathbf{x}) = \mathbf{w}_o^T\mathbf{x} + b_o = r\,\|\mathbf{w}_0\|$$

или

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}_o\|}. \tag{6.5}$$

В частности, расстояние от начала координат (т.е. $\mathbf{x} = \mathbf{0}$) до оптимальной гиперплоскости равно $b_o/\|\mathbf{w}_o\|$. Если $b_o>0$, то начало координат находится с положительной стороны оптимальной гиперплоскости, если же $b_o<0$ — то с отрицательной. Геометрическая интерпретация этих алгебраических результатов представлена на рис. 6.2.

Требуется найти параметры $\mathbf{w}_o$ и b_o оптимальной гиперплоскости на основе данного множества примеров $\mathbf{T}=\{(\mathbf{x}_i, d_i)\}$. В свете результатов, показанных на рис. 6.2, видно, что пара $(\mathbf{w}_o, b_o)$ должна удовлетворять следующим ограничениям:

$$\begin{aligned} \mathbf{w}_o^T\mathbf{x}_i + b_o &\geq 1 \quad \text{для } d_i = +1, \\ \mathbf{w}_o^T\mathbf{x}_i + b_o &\leq -1 \text{ для } d_i = -1. \end{aligned} \tag{6.6}$$

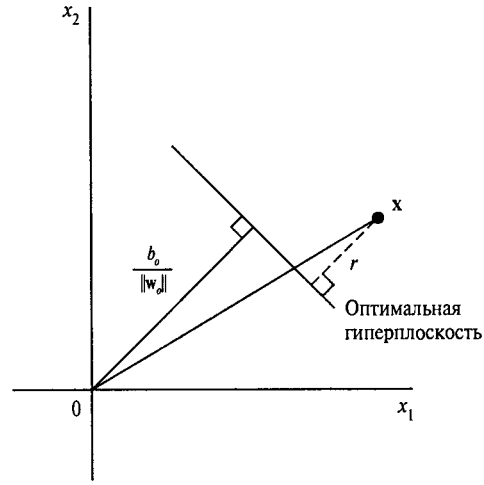


Рис. 6.2. Геометрическая интерпретация алгебраических расстояний от точек до оптимальной гиперплоскости (двумерный случай)

Заметим, что если выполняется уравнение (6.2), то множества являются линейно-разделимыми. В этом случае значения $\mathbf{w}_o$ и b_o можно масштабировать так, чтобы выполнялось условие (6.6). Такая операция масштабирования не влияет на соотношение (6.3).

Конкретные точки $(\mathbf{x}_i, d_i)$, для которых первое и второе ограничения выполняются со знаком равенства, называются *опорными векторами* (support vector). Отсюда и получили свое название машины опорных векторов. Эти векторы играют решающую роль в работе обучаемых машин. А именно, опорные векторы являются теми точками данных, которые лежат ближе всего к поверхности решений, и, таким образом, являются самыми сложными для классификации. Как таковые они лучше всего указывают на оптимальное размещение поверхности решений.

Рассмотрим опорный вектор $\mathbf{x}^{(s)}$, для которого $d^{(s)} = +1$. Тогда, по определению,

$$g(\mathbf{x}^{(s)}) = \mathbf{w}_o^T \mathbf{x}^{(s)} \mp b_o = \mp 1 \text{ для } d^{(s)} = \mp 1. \tag{6.7}$$

Исходя из выражения (6.5), *алгебраическое расстояние* (algebraic distance) от опорного вектора $\mathbf{x}^{(s)}$ до оптимальной гиперплоскости равно

$$r = \frac{g(\mathbf{x}^{(s)})}{\|\mathbf{w}_o\|} = \begin{cases} \frac{1}{\|\mathbf{w}_o\|}, & \text{если } d^{(s)} = +1, \\ -\frac{1}{\|\mathbf{w}_o\|}, & \text{если } d^{(s)} = -1, \end{cases} \tag{6.8}$$

где знак "плюс" означает расположение точки $\mathbf{x}^{(s)}$ с положительной стороны от оптимальной гиперплоскости, а знак "минус" — с отрицательной. Пусть ρ — оптимальное значение *границы разделения* между двумя классами, составляющими множество примеров $\mathbf{T}$. Тогда из уравнения (6.8) следует, что

$$\rho = 2r = \frac{2}{\|\mathbf{w}_o\|}. \tag{6.9}$$

Соотношение (6.9) означает, что максимизация границы разделения между классами эквивалентна минимизации Евклидовой нормы вектора $\mathbf{w}$.

Подводя итог, отметим, что оптимальная гиперплоскость, определяемая уравнением (6.3), является *единственной* (unique) в том смысле, что оптимальный вектор весов $\mathbf{w}_o$ обеспечивает максимально возможное разделение между положительными и отрицательными примерами. Такое оптимальное состояние достигается с помощью минимизации Евклидовой нормы вектора весов $\mathbf{w}$.

Квадратичная оптимизация и поиск оптимальной гиперплоскости

Нашей целью является разработка вычислительно эффективной процедуры использования обучающего множества $\mathbf{T}=\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ для поиска оптимальной гиперплоскости, удовлетворяющей следующему ограничению:

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \text{ для } i = 1, 2, \ldots, N. \tag{6.10}$$

Это ограничение объединяет оба неравенства (6.6), если вместо $\mathbf{w}_o$ используется $\mathbf{w}$.

Задачу условной оптимизации, которую необходимо решить, можно описать следующим образом.

Для данного обучающего множества $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ *найти оптимальные значения вектора весовых коэффициентов* $\mathbf{w}$ *и порога* b, *удовлетворяющие условию*

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \text{ для } i = 1, 2, \ldots, N$$

и минимизирующие функцию стоимости

$$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}.$$

Масштабирующий множитель $1/2$ включен для удобства выкладок. Такая задача условной оптимизации называется *прямой задачей* и удовлетворяет следующим условиям.

- Функция стоимости $\Phi(\mathbf{w})$ является выпуклой[1].
- Ограничения линейны по $\mathbf{w}$.

---

[1] Пусть $\mathbf{C}$ – некоторое подмножество $\Re^m$. Это подмножество называется *выпуклым*, если

$$\alpha x + (1 - \alpha)y \in \mathbf{C} \text{ для всех } (x, y) \in \mathbf{C} \text{ и } \alpha \in [0, 1].$$

Функция $f : \mathbf{C} \longrightarrow \Re$ называется *выпуклой*, если

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y) \text{ для всех } (x, y) \in \mathbf{C} \text{ и } \alpha \in [0, 1].$$

Следовательно, эту задачу можно решить с помощью *метода множителей Лагранжа* (Method of Lagrange multipliers) [124].

Построим сначала функцию Лагранжа:

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1], \tag{6.11}$$

где дополнительные неотрицательные переменные α_i — это так называемые *множители Лагранжа* (Lagrange multiplier). Решение задачи условной оптимизации определяется *седловой точкой* (saddle point) функции Лагранжа $J(\mathbf{w}, b, \alpha)$, которую необходимо *минимизировать* по $\mathbf{w}$ и b, одновременно *максимизируя* по α. Таким образом, дифференцируя $J(\mathbf{w}, b, \alpha)$ по $\mathbf{w}$ и b и приравнивая результат к нулю, получим следующие *условия оптимальности*.

Условие 1: $\frac{\partial J(\mathbf{w},b,\alpha)}{\partial \mathbf{w}} = \mathbf{0}$.

Условие 2: $\frac{\partial J(\mathbf{w},b,\alpha)}{\partial b} = 0$.

Применяя условие оптимальности 1 к функции Лагранжа (6.11), после перестановки слагаемых получим:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \mathbf{x}_i. \tag{6.12}$$

Применяя условие оптимальности 2 к функции Лагранжа (6.11), получим:

$$\sum_{i=1}^{N} \alpha_i d_i = 0. \tag{6.13}$$

Вектор решения $\mathbf{w}$ определяется в терминах расширения, включающего N примеров обучения. Это решение является единственным благодаря выпуклости Лагранжиана, чего нельзя сказать о коэффициентах Лагранжа α_i.

Важно также заметить, что в седловой точке для каждого множителя Лагранжа α_i произведение этого множителя на соответствующее ограничение сходится к нулю, т.е.

$$\alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1] = 0 \text{ для } i = 1, 2, \ldots, N. \tag{6.14}$$

Таким образом, *ненулевые* значения могут иметь только те множители, которые в точности удовлетворяют условию (6.14). Это свойство следует из *условия Куна–Такера* (Kuhn-Tucker condition), сформулированного в теории оптимизации [124], [298].

Как уже отмечалось, решение прямой задачи предполагает выпуклость функции стоимости и линейность ограничений. Для такой задачи условной оптимизации можно сформулировать так называемую *двойственную задачу* (dual problem), которая будет иметь то же оптимальное решение, что и прямая, однако оптимальность ее

решения будет обеспечиваться множителями Лагранжа. В частности, можно сформулировать следующую *теорему двойственности* (duality theorem) [124].

(а) *Если прямая задача имеет оптимальное решение, то двойственная задача также имеет оптимальное решение, при этом соответствующие оптимальные решения совпадают.*

(б) *Для того чтобы* $\mathbf{w}_o$ *было оптимальным решением прямой задачи, а* α_o *— оптимальным решением двойственной, необходимо и достаточно, чтобы* $\mathbf{w}_o$ *было допустимым решением прямой задачи и*

$$\Phi(\mathbf{w}_o) = J(\mathbf{w}_o, b_o, \alpha_o) = \min_{\mathbf{w}} J(\mathbf{w}, b_o, \alpha_o). \tag{6.15}$$

Чтобы сформулировать двойственную задачу для рассмотренной прямой, раскроем выражение (6.11):

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b\sum_{i=1}^{N} \alpha_i d_i + \sum_{i=1}^{N} \alpha_i.$$

Третье слагаемое в правой части () равно нулю вследствие условия оптимальности (6.13). Далее, из (6.12) получим:

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \mathbf{w}^T \mathbf{x}_i = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j.$$

Следовательно, представив целевую функцию как $J(\mathbf{w},b,\alpha) = Q(\alpha)$, соотношение () можно переформулировать следующим образом:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j, \tag{6.16}$$

где α_i — неотрицательны.

Теперь можно сформулировать двойственную задачу.

Для данного обучающего множества $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ *найти множители Лагранжа* $\{\alpha_i\}_{i=1}^N$*, максимизирующие целевую функцию*

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j$$

при следующих ограничениях

1. $\sum_{i=1}^{N} \alpha_i d_i = 0,$
2. $\alpha_i \geq 0$ для $i = 1, 2, \ldots, N.$

Обратите внимание, что двойственная задача целиком описывается в терминах данных обучения. Более того, максимизируемая функция $Q(\alpha)$ зависит *только* от входных образов, представленных в виде множества скалярных произведений $\{\mathbf{x}_i^T \mathbf{x}_j\}_{(i,j)=1}^N$.

Определив оптимальные множители Лагранжа $\alpha_{o,i}$, с помощью соотношения (6.12) можно вычислить оптимальный вектор весовых коэффициентов $\mathbf{w}_o$:

$$\mathbf{w}_o = \sum_{i=1}^{N} \alpha_{o,i} d_i \mathbf{x}_i. \tag{6.17}$$

Для вычисления оптимального значения порога b_o можно использовать полученное значение $\mathbf{w}_o$ и выражение (6.7):

$$b_o = 1 - \mathbf{w}_o^T \mathbf{x}^{(s)} \text{ для } d^{(s)} = 1. \tag{6.18}$$

Статистические свойства оптимальной гиперплоскости

Из описанной в главе 2 теории статистического обучения следует, что VC-размерность обучаемой машины определяет способ использования вложенной структуры аппроксимирующих функций. Напомним также, что VC-размерность множества разделяющих гиперплоскостей в пространстве размерности m равна $m + 1$. Тем не менее, чтобы применить метод минимизации структурного риска (см. главу 2), требуется построить такое множество разделяющих гиперплоскостей переменной VC-размерности, которое одновременно минимизирует эмпирический риск (т.е. ошибку классификации на обучающем множестве) и VC-размерность. В машине опорных векторов структура множества разделяющих гиперплоскостей определяется Евклидовой нормой вектора весовых коэффициентов $\mathbf{w}$. В частности, можно сформулировать следующую теорему [1084], [1085].

Пусть D — диаметр наименьшего шара, содержащего все входные векторы $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. VC-размерность h множества оптимальных гиперплоскостей, описываемых уравнением

$$\mathbf{w}_o^T \mathbf{x} + b = 0,$$

ограничена сверху величиной

$$h \leq \min\left\{ \left\lceil \frac{D^2}{\rho^2} \right\rceil, m_o \right\} + 1, \tag{6.19}$$

где символ $\lceil . \rceil$ означает наименьшее целое число, большее или равное заключенному в скобки значению; ρ — граница разделения, равная $2/||\mathbf{w}_o||$; m_0 — размерность входного пространства.

Из этой теоремы следует, что VC-размерностью (т.е. сложностью) оптимальной гиперплоскости можно управлять независимо от размерности входного пространства с помощью правильного выбора границы разделения ρ.

Предположим, существует некоторая вложенная структура, описываемая в терминах разделяющих гиперплоскостей следующим образом:

$$S_k = \{\mathbf{w}^T\mathbf{x} + b : \|\mathbf{w}\|^2 \leq c_k\}, \quad k = 1, 2, \ldots. \tag{6.20}$$

С помощью верхней границы VC-размерности h, определяемой неравенством (6.19), вложенную структуру (6.20) можно переписать в эквивалентной форме в терминах границ разделения:

$$S_k = \left\{ \left\lceil \frac{r^2}{\rho^2} \right\rceil + 1 : \rho^2 \geq a_k \right\}, \quad k = 1, 2, \ldots, \tag{6.21}$$

где a_k и c_k — константы.

Из главы 2 следует, что для обеспечения хорошей обобщающей способности, в соответствии с принципом минимизации структурного риска, необходимо выбрать структуру с наименьшими VC-размерностью и ошибкой обучения. Из выражений (6.19) и (6.21) видно, что это требование можно удовлетворить с помощью оптимальной гиперплоскости (т.е. с помощью разделяющей гиперплоскости с наибольшей границей разделения ρ). Либо, в свете выражения (6.9), необходимо использовать оптимальный вектор весовых коэффициентов $\mathbf{w}_o$ с минимальной Евклидовой нормой. Таким образом, выбор оптимальной гиперплоскости как поверхности решений для множества линейно-разделимых множеств не только интуитивно удовлетворяет, но и полностью соответствует принципу минимизации структурного риска машины опорных векторов.

6.3. Оптимальная гиперплоскость для неразделимых образов

До сих пор в этой главе рассматривались только линейно-разделимые образы. Теперь сконцентрируем внимание на более сложном случае — неразделимых множествах. При таком наборе данных обучения невозможно построить разделяющую гиперплоскость, полностью исключающую ошибки классификации. Тем не менее мы попытаемся сконструировать оптимальную гиперплоскость, которая сводит к минимуму вероятность таких ошибок на множестве обучающих примеров.

Граница разделения классов считается *мягкой* (soft), если некоторая точка $(\mathbf{x}_i, d_i)$ нарушает следующее условие (см. (6.10)):

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, i = 1, 2, \ldots, N.$$

Такое нарушение может иметь две формы.

- Точка $(\mathbf{x}_i, d_i)$ попадает в область разделения с нужной стороны поверхности решений (рис. 6.3, *а*).
- Точка попадает в область разделения с другой стороны поверхности решений (рис. 6.3, *б*).

Заметим, что в первом случае классификация будет правильной, а во втором — ошибочной.

Для того чтобы подготовить почву для формального рассмотрения неразделимых образов, введем в определение разделяющей гиперплоскости (поверхности решений) новое множество неотрицательных скалярных переменных $\{\xi_i\}_{i=1}^N$:

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 + \xi_i, i = 1, 2, \ldots, N. \tag{6.22}$$

Величина ξ_i называется *фиктивной переменной* (slack variable), определяющей отклонение точек от идеального состояния линейной разделимости. Если $0 \leq \xi_i \leq 1$, то точка попадает в область границы разделения, но с нужной стороны поверхности решений (см. рис. 6.3, *а*). Если же $\xi_i > 1$, то точка попадает в область с неверной стороны разделяющей гиперплоскости (см. рис. 6.3, *б*). Опорные векторы являются теми точками, которые точно удовлетворяют неравенству (6.22) при $\xi_i > 0$. Заметим, что если образ с $\xi_i > 0$ выбросить из обучающего множества, то поверхность решения изменится. Таким образом, опорные векторы определяются одинаково как в случае разделимых, так и в случае неразделимых образов.

Задача сводится к поиску разделяющей гиперплоскости, минимизирующей ошибку классификации на множестве обучающих примеров. Это можно обеспечить, минимизируя функционал

$$\Phi(\xi) = \sum_{i=1}^{N} I(\xi_i - 1)$$

по вектору весовых коэффициентов $\mathbf{w}$ при условии (6.22) и ограничениях на $||\mathbf{w}||^2$. Здесь $I(\xi)$ — *функция индикатора*, определяемая следующим образом:

$$I(\xi) = \begin{cases} 0, & \xi \leq 0, \\ 1, & \xi > 0. \end{cases}$$

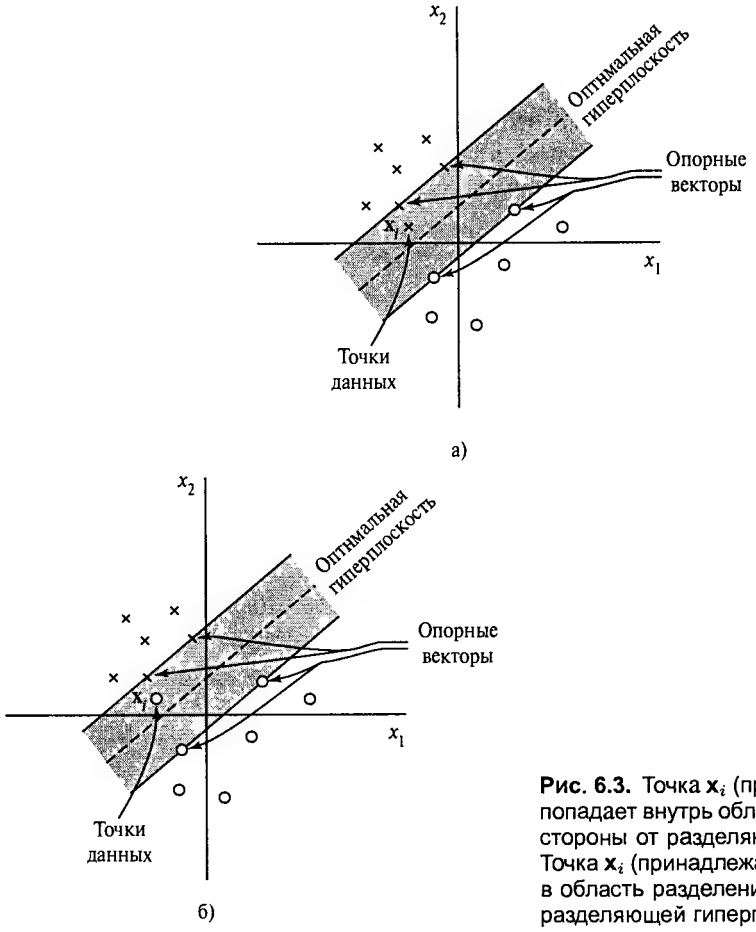


Рис. 6.3. Точка $\mathbf{x}_i$ (принадлежащая классу $\mathbf{C}_1$) попадает внутрь области разделения с нужной стороны от разделяющей гиперплоскости (а). Точка $\mathbf{x}_i$ (принадлежащая классу $\mathbf{C}_2$) попадает в область разделения с неверной стороны от разделяющей гиперплоскости (б)

К сожалению, минимизация $\Phi(\xi)$ по $\mathbf{w}$ является невыпуклой NP-полной[2] задачей минимизации.

---

[2] С точки зрения вычислительной сложности алгоритмы можно разделить на два класса.

Алгоритмы с полиномиальным временем. Время их выполнения описывается полиномиальной функцией от размерности задачи. Например, алгоритм быстрого преобразования Фурье, используемый в спектральном анализе, является алгоритмом полиномиальной сложности, так как требует времени выполнения $n\log n$, где n — размерность задачи.

Алгоритмы с экспоненциальным временем. Время их выполнения пропорционально экспоненциальной функции от размерности задачи. Например, выполнение алгоритма экспоненциальной сложности может потребовать времени 2^n, где n — размерность задачи.

С таких позиций алгоритмы с полиномиальным временем выполнения можно рассматривать как "эффективные", а алгоритмы с экспоненциальным — как "неэффективные".

На практике существует множество вычислительных задач, для которых пока не предложено эффективных алгоритмов. Многие из таких трудноразрешимых задач относят к так называемому классу *NP-полных задач* (NP-complete). Аббревиатура NP означает nondeterministic polynomial (недетерминировано полиномиальные).

Более полное описание NP-полных задач содержится в [208], [211], [338].

Чтобы обеспечить математическую трактовку такой задачи минимизации, аппроксимируем функционал $\Phi(\xi)$ следующим образом:

$$\Phi(\xi) = \sum_{i=1}^{N} \xi_i.$$

Более того, упростим вычисления, записав минимизируемый функционал относительно вектора весов $\mathbf{w}$:

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N} \xi_i. \tag{6.23}$$

Как и ранее, минимизация первого слагаемого в (6.23) связана с минимизацией VC-размерности машины опорных векторов. Что же касается второго слагаемого, то оно представляет верхнюю границу количества ошибок тестирования. Таким образом, представление функции стоимости $\Phi(\mathbf{w}, \xi)$ в (6.23) полностью согласуется с принципом минимизации структурного риска.

Параметр C обеспечивает компромисс между сложностью машины и количеством неразделимых точек. Его можно рассматривать как некоторую форму "параметра регуляризации". Параметр C выбирается пользователем. Его можно выбрать двумя способами.

- *Экспериментально* на основе множества обучения/тестирования.
- *Аналитически*, оценивая VC-размерность с помощью (6.19) и используя границы эффективности обобщения машины на основе VC-размерности.

В любом случае функционал $\Phi(\mathbf{w}, \xi)$ оптимизируется по $\mathbf{w}$ и $\{\xi_i\}_{i=1}^N$ при условиях (6.22) и $\xi_i \geq 0$. При этом квадрат нормы $\mathbf{w}$ трактуется как значение, минимизируемое с учетом неразделимости точек, а не как ограничение, накладываемое на минимизацию количества неразделимых точек.

Сформулированная таким образом задача оптимизации для неразделимых множеств в качестве частного случая включает в себя задачу оптимизации линейно-разделимых множеств. В частности, установив значение ξ равным нулю для всех i, уравнения (6.22) и (6.23) можно свести к виду, соответствующему линейно-разделяемым множествам.

Теперь можно формально определить прямую задачу для неразделимых множеств.

Для данного обучающего множества $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ *найти оптимальные значения вектора весов* $\mathbf{w}$ *и порога* b, *удовлетворяющие ограничениям*

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, N,$$
$$\xi_i \geq 0 \text{ для всех } i,$$

такие, что вектор $\mathbf{w}$ *совместно с* ξ_i *минимизируют функционал стоимости*

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i,$$

где C *— задаваемый пользователем положительный параметр.*

Используя метод множителей Лагранжа и выполняя преобразования, аналогичные приведенным в разделе 6.2, можно сформулировать двойственную задачу для неразделимых множеств.

Для данного обучающего множества $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$ *найти множители Лагранжа* $\{\alpha_i\}_{i=1}^{N}$, *максимизирующие целевую функцию*

$$Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j$$

при ограничениях

- $\sum_{i=1}^{N}\alpha_i d_i = 0,$
- $0 \le \alpha_i \le C$ для $i = 1, 2, \ldots, N,$

где C *— задаваемый пользователем положительный параметр.*

Обратите внимание, что в двойственной задаче не фигурируют ни фиктивные переменные ξ_i, ни их множители Лагранжа. Таким образом, двойственная задача для неразделимых множеств аналогична более простой задаче для линейно-разделимых образов во всем, за исключением одного, но крайне важного отличия. В обоих случаях максимизируется одна и та же целевая функция $Q(\alpha)$. Однако в случае неразделимых классов ограничение $\alpha_i \ge 0$ заменено более строгим — $0 \le \alpha_i \le C$. За исключением этого изменения, задача условной оптимизации для неразделимых образов и вычисления оптимальных значений вектора весов $\mathbf{w}$ и порога b не отличается от случая линейно-разделимых множеств. Заметим также, что опорные векторы вычисляются точно таким же способом, как и ранее.

Оптимальное значение вектора весовых коэффициентов $\mathbf{w}$ определяется по формуле

$$\mathbf{w}_o = \sum_{i=1}^{N_s}\alpha_{o,i}d_i\mathbf{x}_i, \tag{6.24}$$

где N_s — количество опорных векторов. Процедура определения оптимального значения порога также аналогична описанной ранее. В частности, условие Куна–Такера

определяется следующим образом:

$$\alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i] = 0, \quad i = 1, 2, \ldots, N \tag{6.25}$$

и

$$\mu_i\xi_i = 0, i = 1, 2, \ldots, N. \tag{6.26}$$

Уравнение (6.25) является полным эквивалентом (6.14), в котором единичное слагаемое заменено выражением $(1 - \xi_i)$. Как и в (6.14), μ_i — множители Лагранжа, которые были введены для обеспечения неотрицательности переменных ξ_i для всех i. В седловой точке производная функции Лагранжа по ξ_i в прямой задаче равна нулю, откуда следует, что

$$\alpha_i + \mu_i = C. \tag{6.27}$$

Объединяя (6.26) и (6.27), мы видим, что

$$\xi_i = 0, \text{ если } \alpha_i < C. \tag{6.28}$$

Выбирая любую точку $(\mathbf{x}_i, d_i)$ из обучающего множества, для которой $0 < \alpha_{o.i} < C$, и, таким образом, $\xi_i = 0$, и используя (6.25), можно вычислить оптимальный порог b_o. Однако с вычислительной точки зрения порог b_o лучше выбирать как среднее значение по всем точкам обучающего множества [165].

6.4. Как создать машину опорных векторов для задачи распознавания образов

Ознакомившись со способом нахождения оптимальной гиперплоскости для неразделимых множеств, можно формально описать процесс создания машины опорных векторов для задачи распознавания образов.

Идея машины опорных векторов[3] базируется на двух математических операциях, приведенных ниже (рис. 6.4).

[3] Идея представления ядра в виде скалярного произведения впервые была использоваиа в [11], [12] при описании метода потенциальных функций, который являлся предтечей сетей на основе радиальных базисных функций. Примерио в то же время в [1089] была предложена идея оптимальных гиперплоскостей. Сочетание этих двух мощных концепций при создании машины опориых векторов впервые было описаио в [141, [212], [1086]. Полный математический базис машины опорных векторов был изложен в [1085] и позднее в расширенной форме представлен в [1084].

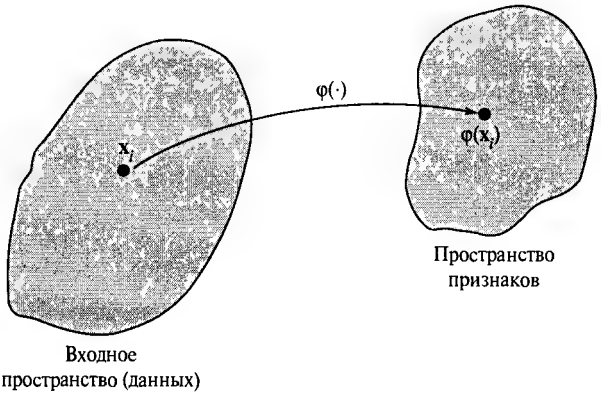


Рис. 6.4. Нелинейное отображение $\varphi(\cdot)$ входного пространства в пространство признаков

1. Нелинейное *отображение* входного вектора в *пространство признаков* (feature space) более высокой размерности.
2. Построение оптимальной гиперплоскости для разделения признаков, полученных в п. 1.

Рассмотрим смысл каждой из этих двух операций.

Операция 1 выполняется в соответствии с теоремой Ковера (Cover) о разделимости образов (см. главу 5). Рассмотрим входное пространство, состоящее из *нелинейно-разделимых образов* (nonlinearly separated patterns). Согласно теореме Ковера, такое многомерное пространство можно преобразовать в новое пространство признаков, в котором с высокой вероятностью образы станут линейно-разделимыми, если выполняются два условия. Во-первых, преобразование должно быть нелинейным. Во-вторых, размерность пространства признаков должна быть достаточно большой. Эти два условия включены в операцию 1. Однако заметим, что в теореме Ковера ничего не говорится об оптимальности разделяющих гиперплоскостей. Только с помощью оптимальных гиперплоскостей можно минимизировать VC-размерность и добиться хорошего обобщения.

Этот вопрос и составляет сущность второй операции. В частности, в этой операции используется идея построения оптимальной разделяющей гиперплоскости в соответствии с теорией, описанной в разделе 6.3, но с одним фундаментальным отличием: разделяющая гиперплоскость теперь определяется как линейная функция от векторов, взятых из пространства признаков, а не из исходного входного пространства. И, что более важно, построение гиперплоскости осуществляется в соответствии с принципом минимизации структурного риска, который вытекает из теории VC-размерности. Стержнем этого построения является оценка *ядра скалярного произведения* (inner-product kernel).

Ядро скалярного произведения

Пусть $\mathbf{x}$ — некоторый вектор из входного пространства, размерность которого равна m_0. Пусть $\{\varphi_j(\mathbf{x})\}_{j=1}^{m_1}$ — множество нелинейных преобразований из входного пространства в пространство признаков. Размерность пространства признаков обозначим через m_1. Предполагается, что функции $\varphi_j(\mathbf{x})$ определены для всех j. Имея множество нелинейных преобразований, можно определить гиперплоскость, определяющую поверхность решений

$$\sum_{j=1}^{m_1} w_j \varphi_j(\mathbf{x}) + b = 0, \tag{6.29}$$

где $\{w_j\}_{j=1}^{m_1}$ — множество весовых коэффициентов, связывающих пространство признаков с выходным пространством; b — порог.

Это выражение можно упростить:

$$\sum_{j=0}^{m_1} w_j \varphi_j(\mathbf{x}) = 0, \tag{6.30}$$

считая, что $\varphi_0(\mathbf{x}) = 1$ для всех $\mathbf{x}$. При этом w_0 представляет собой порог b. Уравнение (6.30) определяет поверхность решений, вычисленную в пространстве признаков в терминах линейных весовых коэффициентов машины. Величина $\varphi_j(\mathbf{x})$ представляет собой входной сигнал, приходящийся на вес w_j в пространстве признаков. Определим вектор

$$\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_{m1}(\mathbf{x})]^T, \tag{6.31}$$

где по определению

$$\varphi_0(x) = 1 \text{ для всех } \mathbf{x}. \tag{6.32}$$

В результате *вектор признаков* (feature vector) $\boldsymbol{\varphi}(\mathbf{x})$ представляет собой "образ", индуцированный в пространстве признаков при предъявлении вектора входного сигнала $\mathbf{x}$ (см. рис. 6.4). Таким образом, в терминах этого образа можно определить поверхность решений в компактной форме:

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = 0. \tag{6.33}$$

Адаптируя (6.12) к задаче линейного разделения векторов в пространстве признаков, можно записать:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \varphi(\mathbf{x}_i), \tag{6.34}$$

где *вектор признаков* (feature vector) $\varphi(\mathbf{x}_i)$ соответствует входному вектору $\mathbf{x}_i$ в i-м примере.

Подставляя выражение (6.34) в (6.33), можно определить поверхность решений в пространстве признаков следующим образом:

$$\sum_{i=1}^{N} \alpha_i d_i \varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}_i) = 0. \tag{6.35}$$

Здесь терм $\varphi^T(\mathbf{x}_i)\varphi(\mathbf{x})$ представляет собой скалярное произведение двух векторов, индуцированных в пространстве признаков для входного вектора $\mathbf{x}$ и примера $\mathbf{x}_i$. Исходя из этого, можно ввести понятие *ядра скалярного произведения* (inner-product kernel), обозначаемого как $K(\mathbf{x}, \mathbf{x}_i)$:

$$K(\mathbf{x}, \mathbf{x}_i) = \varphi^T(\mathbf{x})\varphi(\mathbf{x}_i) = \sum_{j=0}^{m_1} \varphi_j(\mathbf{x})\varphi_j(\mathbf{x}_i), \qquad i = 1, 2, \ldots, N. \tag{6.36}$$

Из этого определения видно, что ядро скалярного произведения является *симметричной* функцией своих аргументов, т.е. для всех i выполняется соотношение

$$K(\mathbf{x}, \mathbf{x}_i) = K(\mathbf{x}_i, \mathbf{x}). \tag{6.37}$$

Ядро скалярного произведения $K(\mathbf{x}, \mathbf{x}_i)$ можно использовать для построения оптимальной гиперплоскости в пространстве признаков, не представляя его в явном виде. Это отчетливо видно из подстановки (6.36) в (6.35), в результате которой оптимальную гиперплоскость можно определить следующим образом:

$$\sum_{i=1}^{N} \alpha_i d_i K(\mathbf{x}, \mathbf{x}_i) = 0. \tag{6.38}$$

Теорема Мерсера

Формула (6.36) для ядра скалярного произведения $K(\mathbf{x}, \mathbf{x}_i)$ является важным частным случаем *теоремы Мерсера* (Mercer's theorem) из функционального анализа. Эта теорема формулируется следующим образом [217], [728].

Пусть $K(\mathbf{x}, \mathbf{x}')$ *— непрерывное симметричное ядро, определенное на закрытом интервале* $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$ *(это же касается и* $\mathbf{x}'$*). Ядро может быть представлено в виде ряда*

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \varphi_i(\mathbf{x}) \varphi_i(\mathbf{x}') \tag{6.39}$$

с положительными коэффициентами $\lambda_i > 0$ *для всех* i*. Для обеспечения корректности этого разложения и его абсолютной и равномерной сходимости необходимо и достаточно выполнение условия*

$$\int_{\mathbf{b}}^{\mathbf{a}} \int_{\mathbf{b}}^{\mathbf{a}} K(\mathbf{x}, \mathbf{x}') \psi(\mathbf{x}) \psi(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0$$

для всех $\psi(\cdot)$*, для которых*

$$\int_{\mathbf{b}}^{\mathbf{a}} \psi^2(\mathbf{x}) d\mathbf{x} < \infty.$$

Функции $\varphi_i(\mathbf{x})$ называются *собственными функциями* (eigenfunction) разложения, а числа λ_i — ее *собственными значениями* (eigenvalue). Из свойства положительности всех собственных чисел вытекает *положительная определенность* (positive definite) ядра $K(\mathbf{x}, \mathbf{x}')$.

В свете теоремы Мерсера можно сделать следующие наблюдения.

- Для $\lambda_i \neq 1$ i-й образ $\sqrt{\lambda_i}\varphi_i(\mathbf{x})$, индуцированный в пространстве признаков входным вектором $\mathbf{x}$, является собственной функцией разложения.
- Теоретически размерность пространства признаков (т.е. количество собственных функций и собственных значений) может быть бесконечно большой.

Теорема Мерсера позволяет лишь определить, является ли ядро-кандидат ядром скалярного произведения в некотором пространстве, и, таким образом, можно ли его использовать в машине опорных векторов. Однако в ней ничего не говорится о том, как строить функции $\varphi_i(\mathbf{x})$. Это необходимо делать самостоятельно.

Из соотношения (6.23) видно, что машина опорных векторов в *неявном* виде включает некоторую форму регуляризации. В частности, использование ядра $K(\mathbf{x}, \mathbf{x}')$, определенного в соответствии с теоремой Мерсера, соответствует регуляризации с оператором $\mathbf{D}$, таким, что это ядро является функцией Грина оператора $\mathbf{D}\tilde{\mathbf{D}}$, где $\tilde{\mathbf{D}}$ — оператор, сопряженный с $\mathbf{D}$ [1004]. Теория регуляризации рассматривалась в главе 5.

Оптимальная архитектура машины опорных векторов

Разложение ядра скалярного произведения $K(\mathbf{x}, \mathbf{x}')$ (6.36) позволяет построить поверхность решений, которая во входном пространстве является нелинейной, однако *ее образ в пространстве признаков является линейным.* С помощью такого разложения можно определить двойственную задачу условной оптимизации для машины опорных векторов.

Для данного множества обучающих примеров $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ *найти множители Лагранжа* $\{\alpha_i\}_{i=1}^N$, *которые максимизируют целевую функцию*

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{6.40}$$

при ограничениях

1) $\sum_{i=1}^{N} \alpha_i d_i = 0,$

2) $0 \leq \alpha_i \leq C$ *для* $i = 1, 2, \ldots, N,$

где C *— положительный параметр, задаваемый пользователем.*

Заметим, что первое ограничение вытекает из оптимизации Лагранжиана $Q(\alpha)$ по пороговому значению $b = w_0$ для $\varphi_0(\mathbf{x}) = 1$. Сформулированная таким образом двойственная задача имеет тот же вид, что и для линейно-неразделимых множеств, рассмотренных в разделе 6.3, за исключением того факта, что используемое там скалярное произведение $\mathbf{x}_i^T \mathbf{x}_j$ заменено ядром скалярного произведения $K(\mathbf{x}_i, \mathbf{x}_j)$. Ядро $K(\mathbf{x}_i, \mathbf{x}_j)$ можно рассматривать как элемент ij симметричной матрицы $\mathbf{K}$ размерности $N \times N$:

$$\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{(i,j)=1}^{N}. \tag{6.41}$$

Определив оптимальные значения множителей Лагранжа $\alpha_{o,i}$, с помощью адаптированной к нашему случаю формулы (6.17) можно найти соответствующее оптимальное значение вектора $\mathbf{w}_o$ весов, связывающего пространство признаков с выходным пространством. В частности, понимая, что образ $\varphi(\mathbf{x}_i)$ выступает в роли входного сигнала для вектора весов $\boldsymbol{w}$, можно определить $\mathbf{w}_o$ как

$$\mathbf{w}_o = \sum_{i=1}^{N} \alpha_{o,i} d_i \varphi(\mathbf{x}_i), \tag{6.42}$$

где $\varphi(\mathbf{x}_i)$ — образ, индуцированный в пространстве признаков входным вектором $\mathbf{x}_i$. Заметим, что первый компонент вектора $\mathbf{w}_o$ представляет собой оптимальный порог b_o.

ТАБЛИЦА 6.1. Ядра скалярных произведений

Тип машины опорных векторов	*Ядро скалярного произведения* $K(\mathbf{x},\mathbf{x}_i), i = 1, 2, \ldots, N$	*Комментарии*
Полиномиальная машина обучения	$(\mathbf{x}^T\mathbf{x}_i+1)^p$	Степень p задается *априори* пользователем
Сети на основе радиальных базисных функций	$\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}_i\|^2\right)$	Ширина σ^2, общая для всех ядер, определяется *априори* пользователем
Двухслойный персептрон	$\tanh(\beta_0\mathbf{x}^T\mathbf{x}_i+\beta_1)$	Теорема Мерсера выполняется только для некоторых значений β_0 и β_1

Примеры машин опорных векторов

Ядро $K(\mathbf{x}, \mathbf{x}_i)$ должно удовлетворять теореме Мерсера. В пределах этого условия имеется некоторая свобода выбора. В табл. 6.1 приведены ядра скалярных произведений для трех самых распространенных типов машин опорных векторов: полиномиальной машины обучения, сетей на основе радиальных базисных функций и двухслойного персептрона. При этом важно заметить следующее.

1. Ядра скалярных произведений для полиномиального и радиального типов функций в машинах опорных векторов всегда удовлетворяют теореме Мерсера. В противоположность этому ядра скалярного произведения в машине опорных векторов типа двухслойного персептрона имеют некоторые ограничения (см. последнюю строку в табл. 6.1). Эта запись является явным свидетельством того, что задача определения, удовлетворяет ли конкретное ядро условиям теоремы Мерсера, уже сама по себе является довольно сложной (см. упражнение 6.8).
2. Для всех трех типов машин размерность пространства признаков определяется числом опорных векторов, отобранных из обучающих данных для решения задачи условной оптимизации.
3. Рассматриваемая теория машин опорных векторов не требует эвристики, часто используемой для разработки обычных сетей на основе радиальных базисных функций и многослойных персептронов.

- В машинах опорных векторов на основе радиальных базисных функций количество этих функций и их центров определяется автоматически по числу *опорных* векторов и их значений соответственно.
- В машинах опорных векторов типа двухслойного персептрона количество скрытых нейронов и их весовых коэффициентов определяется автоматически по количеству опорных векторов и их значений соответственно.

На рис. 6.5 показана архитектура машины опорных векторов.

Независимо от того, как реализованы машины опорных векторов, принцип их создания коренным образом отличается от традиционного подхода к созданию многослойного персептрона. При традиционном подходе сложность модели определяется поддержанием числа признаков (т.е. скрытых нейронов) на невысоком уровне. С другой стороны, машины опорных векторов предлагают другой способ создания обучаемой машины, при котором управление сложностью модели не зависит от размерности. Кратко рассмотрим эту идею [1084], [1085].

- *Концептуальная задача.* Размерность (скрытого) пространства признаков преднамеренно увеличивается, чтобы в этом пространстве можно было построить поверхность решений в виде гиперплоскости. Для повышения эффективности обобщения сложность модели подчиняется некоторым ограничениям, накладываемым на конструируемую гиперплоскость, которые проявляются при выборе подмножества примеров обучения в качестве опорных векторов.
- *Вычислительная задача.* Числовая оптимизация в пространствах высокой размерности подвержена так называемому "проклятию размерности". Этой вычислительной проблемы удается избежать при использовании ядра скалярного произведения (определяемого в соответствии с теоремой Мерсера) и решении двойственной задачи условной оптимизации, формулируемой во входном пространстве.

6.5. Пример: задача XOR (продолжение)

Чтобы проиллюстрировать процесс создания машины опорных векторов, вернемся к знакомой нам задаче XOR (исключающего ИЛИ), которая уже упоминалась в главах 4 и 5. В табл. 6.2 приводятся входные векторы и желаемые отклики для всех возможных состояний.

Выберем ядро скалярного произведения в виде [187]

$$K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}^T \mathbf{x}_i)^2. \tag{6.43}$$

Пусть $\mathbf{x}=[x_1, x_2]^T$ и $\mathbf{x}_i = [x_{i1}, x_{i2}]^T$. Тогда ядро скалярного произведения $K(\mathbf{x}, \mathbf{x}_i)$ можно выразить в терминах *одночленов* (monomial) различных степеней:

$$K(\mathbf{x}, \mathbf{x}_i) = 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2}.$$

Исходя из этого, образ входного вектора $\mathbf{x}$, индуцированный в пространстве признаков, может быть представлен следующим образом:

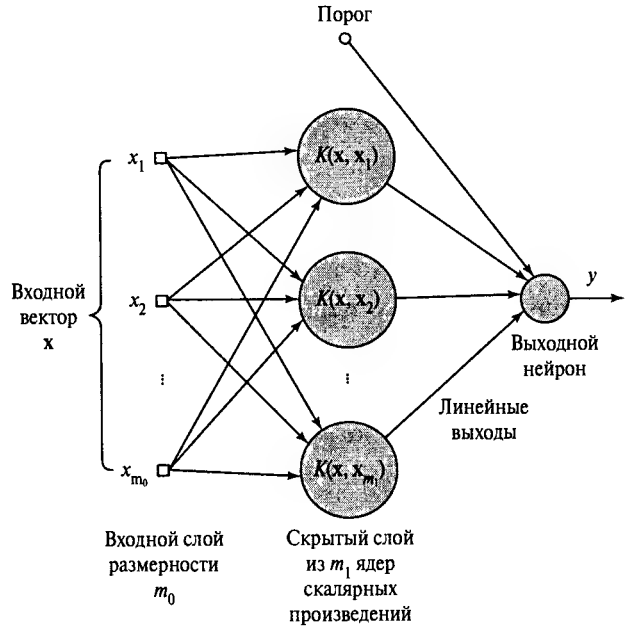


Рис. 6.5. Архитектура машины опорных векторов

$$\varphi(\mathbf{x}) = \left[1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2\right]^T.$$

Аналогично,

$$\varphi(\mathbf{x}_i) = \left[1, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}\right]^T, \quad i = 1, 2, 3, 4.$$

Из равенства (6.41) находим ядро

$$\mathbf{K} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}.$$

Таким образом, целевая функция для двойственной задачи будет иметь следующий вид (см. (6.40)):

$$Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \tfrac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + 9\alpha_2^2 + \\ + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2).$$

ТАБЛИЦА 6.2. Задача XOR

Входной вектор, $\mathbf{x}$	*Желаемый отклик*, d
$(-1,-1)$	-1
$(-1,+1)$	$+1$
$(+1,-1)$	$+1$
$(+1,+1)$	-1

Оптимизируя функцию $Q(\alpha)$ по отношению к множителям Лагранжа, получим следующую систему уравнений:

$$\begin{cases} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1 \\ -\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1 \end{cases}$$

Исходя из этого, оптимальными значениями множителей Лагранжа являются

$$\alpha_{o,1} = \alpha_{o,2} = \alpha_{o,3} = \alpha_{o,4} = \frac{1}{8}.$$

Этот результат означает, что в нашем примере все четыре входных вектора $\{\mathbf{x}_i\}_{i=1}^4$ должны быть выбраны в качестве опорных. Оптимальным значением функции Лагранжа $Q(\alpha)$ будет

$$Q_o(\alpha) = \frac{1}{4}.$$

Следовательно, можно записать

$$\frac{1}{2}\|\mathbf{w}_o\|^2 = \frac{1}{4}$$

или

$$\|\mathbf{w}_o\| = \frac{1}{\sqrt{2}}.$$

Из (6.42) можно найти оптимальный вектор весовых коэффициентов:

$$\mathbf{w}_o = \frac{1}{8}[-\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) + \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)] =$$

$$= \frac{1}{8}\left[-\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix}\right] = \begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

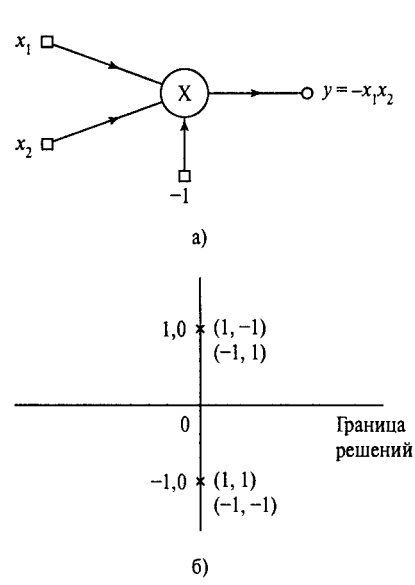


Рис. 6.6. Полиномиальная машина для решения задачи XOR (а); индуцированный образ в пространстве признаков для четырех точек данных при решении задачи XOR (б)

Из первого элемента вектора $\mathbf{w}_o$ видно, что порог b равен нулю.

Таким образом, мы нашли оптимальную гиперплоскость, определяемую согласно (6.33) следующим образом:

$$\mathbf{w}_o^T \boldsymbol{\varphi}(\mathbf{x}) = 0.$$

Подставляя соответствующие значения, получим:

$$\left[0, 0, -\frac{1}{\sqrt{2}}, 0, 0, 0\right] \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = 0,$$

что в сокращенном виде имеет вид

$$-x_1x_2 = 0.$$

Эта полиномиальная форма машины опорных векторов для решения задачи XOR показана на рис. 6.6, *а*. Для $x_1 = x_2 = -1$ и $x_1 = x_2 = +1$ выходной сигнал такой машины равен $y = -1$; а для $x_1 = -1$ и $x_2 = +1$, равно как и для $x_1 = +1$ и $x_2 = -1$, выходной сигнал равен $y = +1$. Таким образом, задача XOR решена (рис. 6.6, *б*).

ТАБЛИЦА 6.3. Сводные результаты эксперимента по классификации двух множеств с помощью машины опорных векторов

	Общая ширина $\sigma^2 = 4$				
	Параметр регуляризации $C = 0,1$				
Вероятность корректной классификации p_C	81,22	81,28	81,55	81,49	81,45
Количество опорных векторов N_s	298	287	283	287	286

6.6. Компьютерное моделирование

В этом компьютерном эксперименте мы снова вернемся к задаче классификации, которая уже упоминалась в главах 4 и 5. Ставится задача классификации двух перекрывающихся двумерных гауссовских распределений, представляемых классами $\mathbf{C}_1$ и $\mathbf{C}_2$. Графики этих двух множеств данных см. на рис. 4.14. Вероятность корректной классификации (оптимальным) байесовским классификатором оценивается как

$$p_c = 81,51\%.$$

В табл. 6.3 приведены результаты компьютерных экспериментов, выполненных на этом множестве данных с помощью машины опорных векторов. Для ядра скалярного произведения использовалась следующая радиальная базисная функция:

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right), \quad i = 1, 2, \ldots, N,$$

где для всех точек множества данных использовалась одна и та же ширина σ^2 =4. Машина обучалась на N =500 точках данных, случайно выбранных из множества, содержащего представителей обоих классов. При этом использовался параметр регуляризации, равный C =0,1.

Результаты, приведенные в табл. 6.3, описывают пять различных реализаций этого эксперимента. Во всех случаях для обучения использовалось 500 точек, а для тестирования — 32000. Вероятность корректной классификации, усредненная по всем этим пяти попыткам, составила 81,40%. Это значение практически равно результату, полученному байесовским классификатором. Тот факт, что в одном эксперименте результат был превзойден, можно списать на погрешность эксперимента.

Практически идеальная классификация, показанная машиной опорных векторов, еще раз подтверждается построением границы решений (рис. 6.7). Здесь показана одна из реализаций машины, выбранная случайным образом. На этом рисунке также показана граница решений байесовского классификатора, представляющая собой круг с центром $\mathbf{x}_c = [-2/3, 0]^T$ и радиусом $r = 2,34$. Рис. 6.6 однозначно подтверждает, что машина опорных векторов способна построить границу решений между классами $\mathbf{C}_1$ и $\mathbf{C}_2$, которая практически так же хороша, как и оптимальная.

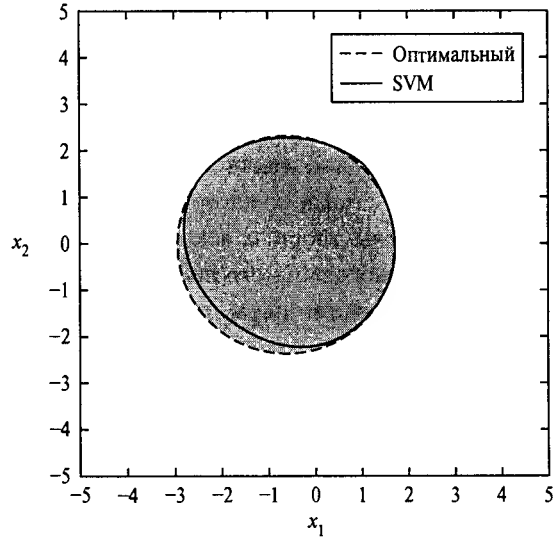


Рис. 6.7. Поверхность решений в компьютерном эксперименте по классификации множеств

Возвращаясь к результатам, представленным в табл. 6.3, обратим внимание на то, что во второй строке представлены размеры пяти различных реализаций машины опорных векторов. Из этих данных видно, что для всех машин в качестве опорных векторов выбиралось приблизительно 60% общего объема обучающих примеров.

В случае неразделимых множеств ошибка обучения приводит к росту числа опорных векторов, что следует также из условия Куна–Такера. В настоящем эксперименте ошибка классификации составила около 20%. Таким образом, для множества из 500 примеров примерно треть опорных векторов была выбрана в связи с ошибками классификации.

Заключительные замечания

Сравнивая результаты этого простого компьютерного эксперимента, проведенного для машины опорных векторов, с результатами, показанными в разделе 4.8 многослойным персептроном, обученным на том же множестве примеров с помощью алгоритма обратного распространения, можно сделать следующие выводы.

Машина опорных векторов обладает встроенной способностью решать задачу классификации множеств, причем получаемое решение *близко к оптимальному*. Более того, она способна добиваться таких результатов без встроенных в конструкцию предварительных знаний о предметной области.

С другой стороны, многослойный персептрон, обучаемый с помощью алгоритма обратного распространения, обеспечивает *вычислительно эффективное* решение задачи классификации множеств. В эксперименте классификации двух множеств с помощью многослойного персептрона с двумя скрытыми нейронами (см. главу 4) мы смогли получить вероятность корректной классификации, близкую к 79,70%.

Подводя итог, мы подчеркнули преимущества каждого из этих подходов к решению задачи классификации. Однако, чтобы сделать объективный вывод, нужно упомянуть и их индивидуальные недостатки. В случае машины опорных векторов близкая к идеальной эффективность была достигнута за счет значительного увеличения вычислительной сложности. С другой стороны, для того чтобы многослойный персептрон, обучаемый по алгоритму обратного распространения, достиг производительности машины опорных векторов, необходимо выполнить два условия: встроить в архитектуру многослойного персептрона знания о конкретной проблемной области и настроить множество параметров, что может быть мучительно тяжело в сложных задачах обучения.

6.7. ε-нечувствительные функции потерь

До сих пор в данной главе мы концентрировали внимание исключительно на использовании машин опорных векторов для решения задач классификации. Теперь же мы займемся вопросами их применения для решения задач нелинейной регрессии. Чтобы подготовить почву для этого исследования, прежде всего рассмотрим вопрос выбора критерия оптимизации, подходящего для этого класса задач.

В главе 4, посвященной многослойным персептронам, и в главе 5, в которой описывались сети на основе радиальных базисных функций, в качестве критерия оптимизации мы использовали квадратичную *функцию потерь* (loss function). Главным принципом при таком выборе было математическое удобство вычислений. Однако система минимизации среднеквадратической ошибки чувствительна к наличию исключений (т.е. наблюдений, выходящих за рамки номинальной модели) и плохо работает для распределений аддитивного шума с длинным "хвостом". Чтобы избежать этих ограничений, необходимо создать *робастную* систему оценивания, которая была бы *нечувствительной к небольшим изменениям в модели.*

Выбрав главной целью робастность, нужно ввести количественную меру робастности, связанную с максимальным снижением производительности при ε-отклонениях от номинальной модели шума. Согласно такой точке зрения, *оптимальная робастная процедура оценивания* (optimal robust estimation procedure) минимизирует максимальное убывание и, таким образом, становится *минимаксной* процедурой [492]. Если аддитивный шум имеет симметричную относительно центра координат функцию плотности вероятности, минимаксная процедура[4] для решения задач нелинейной регрессии в качестве минимизируемой величины использует абсолютную ошибку [493]. Это значит, что функция потерь имеет следующий вид:

[4] Минимаксная теория Габера (Huber) основывается на принципе соседства. Она не является глобальной, поскольку не учитывает асимметричные распределения. Тем не менее эта теория успешно работает в большинстве традиционных задач статистики, в частности в задачах регрессии.

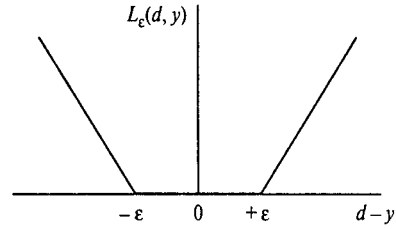


Рис. 6.8. ε-нечувствительная функция потерь

$$L(d,y) = |d-y|, \tag{6.44}$$

где d — желаемый отклик; y — выход системы оценивания.

Чтобы создать машину опорных векторов для аппроксимации желаемого отклика d, можно использовать расширение функции потерь (6.44), впервые предложенное в [1084], [1085], в следующем виде:

$$L_\varepsilon(d,y) = \begin{cases} |d-y| - \varepsilon & \text{для } |d-y| \geq \varepsilon, \\ 0 & \text{в остальных случаях,} \end{cases} \tag{6.45}$$

где ε — наперед заданный параметр.

Функция потерь $L_\varepsilon(d,y)$ называется *ε-нечувствительной функцией потерь* (ε-insensitive loss function). Эта функция равна нулю, если абсолютное значение отклонения выхода системы оцениванияy от желаемого отклика d не превышает ε, и величине отклонения за вычетом ε — в остальных случаях. Функция потерь (6.44) является частным случаем ε-нечувствительной функции потерь при $\varepsilon = 0$. На рис. 6.8 показана зависимость функции $L_\varepsilon(d,y)$ от величины $(d-y)$.

6.8. Машины опорных векторов для задач нелинейной регрессии

Рассмотрим *модель нелинейной регрессии* (nonlinear regressive model), в которой зависимость скаляра d от вектора $\mathbf{x}$ описывается следующим образом:

$$d = f(\mathbf{x}) + v. \tag{6.46}$$

Скалярная нелинейная функция $f(\mathbf{x})$ определяется условным ожиданием $E[D|\mathbf{x}]$ (см. главу 2), где D — случайная переменная, реализация которой обозначается символом d. Аддитивный шум v является статистически независимым от входного вектора $\mathbf{x}$. Функция $f(\cdot)$ и статистика шума v неизвестны. Все, что есть для решения задачи, — это множество обучающих данных $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, где $\mathbf{x}_i$ — значение конкретного входного вектора $\mathbf{x}$; d_i — соответствующее ему значение выхода модели d. Задача состоит в оценке зависимости d от $\mathbf{x}$.

Далее введем оценку значения d, которую обозначим символом y. Она выражается в терминах множества нелинейных базисных функций $\{\varphi_j(\mathbf{x})\}_{j=0}^{m_1}$ следующим образом:

$$y = \sum_{j=0}^{m_1} w_j \varphi_j(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}), \tag{6.47}$$

где

$$\varphi(\mathbf{x}) = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T$$

и

$$\mathbf{w} = [w_0, w_1, \ldots, w_{m_1}]^T.$$

Как и ранее, предполагается, что $\varphi_0(\mathbf{x})$=1, вследствие чего весовой коэффициент w_0 будет равен порогу b. Требуется решить задачу минимизации эмпирического риска

$$R_{\text{emp}} = \frac{1}{N} \sum_{i=1}^{N} L_\varepsilon(d_i, y_i). \tag{6.48}$$

при условии

$$||\mathbf{w}||^2 \leq c_0, \tag{6.49}$$

где c_0 — константа.

В выражении (6.48) используется определенная ранее в (6.45) ε-нечувствительная функция потерь $L_\varepsilon(d, y)$. Эту задачу условной оптимизации можно переформулировать, введя два множества неотрицательных *фиктивных переменных* (slack variable), $\{\xi_i\}_{i=1}^N$ и $\{\xi_i'\}_{i=1}^N$, определяемых следующим образом:

$$d_i - \mathbf{w}^T \varphi(\mathbf{x}_i) \leq \varepsilon + \xi_i, \quad i = 1, 2, \ldots, N, \tag{6.50}$$

$$\mathbf{w}^T \varphi(\mathbf{x}_i) - d_i \leq \varepsilon + \xi_i', \quad i = 1, 2, \ldots, N, \tag{6.51}$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, N, \tag{6.52}$$

$$\xi_i' \geq 0, \quad i = 1, 2, \ldots, N. \tag{6.53}$$

Фиктивные переменные ξ_i и ξ_i' описывают ε-нечувствительную функцию потерь из (6.45). Таким образом, задачу условной оптимизации можно рассматривать как задачу минимизации функционала стоимости

$$\Phi(\mathbf{w}, \xi, \xi') = C \left(\sum_{i=1}^{N} (\xi_i + \xi_i') \right) + \frac{1}{2} \mathbf{w}^T \mathbf{w} \tag{6.54}$$

при ограничениях (6.50)–(6.53). Включая слагаемое $\mathbf{w}^T\mathbf{w}/2$ в функционал $\Phi(\mathbf{w}, \xi, \xi')$ из (6.54), мы избавляемся от необходимости включать в состав ограничений неравенство (6.49). Константа C в (6.54) является параметром, назначаемым пользователем.

Исходя из этого, функцию Лагранжа можно определить следующим образом:

$$J(\mathbf{w},\xi,\xi',\alpha,\alpha',\gamma,\gamma') = C\sum_{i=1}^{N}(\xi+\xi') + \tfrac{1}{2}\mathbf{w}^T\mathbf{w} - \\ -\sum_{i=1}^{N}\alpha_i[\mathbf{w}^T\varphi(\mathbf{x}_i) - d_i + \varepsilon + \xi_i] - \sum_{i=1}^{N}\alpha_i'[d_i - \mathbf{w}^T\varphi(\mathbf{x}_i) + \varepsilon + \xi_i'] - \\ -\sum_{i=1}^{N}(\gamma_i\xi_i + \gamma_i'\xi_i'), \tag{6.55}$$

где α_i и α_i' — множители Лагранжа. Последнее слагаемое в правой части (6.55), содержащее γ_i и γ_i', гарантирует, что условия оптимальности множителей Лагранжа α_i и α_i' предполагают переменную форму. Требуется минимизировать $J(\mathbf{w},\xi,\xi',\alpha,\alpha',\gamma,\gamma')$ по вектору весовых коэффициентов $\mathbf{w}$ и фиктивным переменным ξ и ξ' при одновременной максимизации по α,α', γ и γ'. Выполняя эту оптимизацию, получим:

$$\mathbf{w} = \sum_{i=1}^{N}(\alpha_i - \alpha_i')\varphi(\mathbf{x}_i), \tag{6.56}$$

$$\gamma_i = C - \alpha_i \tag{6.57}$$

и

$$\gamma_i' = C - \alpha_i'. \tag{6.58}$$

Описанная задача оптимизации функционала $J(\mathbf{w},\xi,\xi',\alpha,\alpha',\gamma,\gamma')$ является прямой задачей регрессии. Для того чтобы сформулировать соответствующую двойственную задачу, подставим соотношения (6.56)–(6.58) в (6.55), получив, таким образом, следующий выпуклый функционал (после приведения подобных):

$$Q(\alpha_i,\alpha_i') = \sum_{i=1}^{N}d_i(\alpha_i - \alpha_i') - \varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i') - \\ -\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')K(\mathbf{x}_i,\mathbf{x}_j), \tag{6.59}$$

где $K(\mathbf{x}_i,\mathbf{x}_j)$ — ядро скалярного произведения, определенное согласно теореме Мерсера:

$$K(\mathbf{x}_i,\mathbf{x}_j) = \boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}_j).$$

Таким образом, решение задачи условной оптимизации обеспечивается путем максимизации $Q(\alpha,\alpha')$ по множителям Лагранжа α и α' при новых ограничениях, включающих константу C в определение функции $\Phi(\mathbf{w},\xi,\xi')$ из (6.54).

Теперь можно сформулировать двойственную задачу нелинейной регрессии на основе машины опорных векторов в следующем виде.

Для данного множества примеров обучения $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ *найти множители Лагранжа* $\{\alpha_i\}_{i=1}^N \{\alpha_i'\}_{i=1}^N$, *максимизирующие целевую функцию*

$$Q(\alpha_i, \alpha_i') = \sum_{i=1}^{N} d_i(\alpha_i - \alpha_i') - \varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i') - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_i')(\alpha_j - \alpha_j') K(\mathbf{x}_i, \mathbf{x}_j)$$

при следующих ограничениях:

1. $\sum_{i=1}^{N} (\alpha_i - \alpha_i') = 0,$
2. $0 \le \alpha_i \le C, \quad i = 1, 2, \ldots, N,$
   $0 \le \alpha_i' \le C, \quad i = 1, 2, \ldots, N,$

где C *— константа, задаваемая пользователем.*

Первое ограничение получается из оптимизации Лагранжиана по отношению к порогу $b = w_0$ для $\varphi_0(\mathbf{x}) = 1$. Таким образом, вычислив оптимальные значения для α_i и α_i', можно использовать (6.56) для определения оптимального вектора весов $\mathbf{w}$ для заданного отображения $\varphi(\mathbf{x})$. Обратите внимание, что в решении задачи распознавания только некоторые из коэффициентов разложения (6.56) имеют значения, отличные от нуля. В частности, точки данных, для которых $\alpha_i \neq \alpha_i'$, определяют опорные векторы машины.

Два параметра, ε и C, являются свободными и характеризуют VC-размерность аппроксимирующей функции

$$F(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^{N} (\alpha_i - \alpha_i') K(\mathbf{x}, \mathbf{x}_i). \tag{6.60}$$

Как ε, так и C выбираются пользователем. В концептуальном смысле при выборе ε и C возникают те же вопросы управления сложностью, что и при выборе параметра C в задаче классификации. Однако на практике управление сложностью в задаче регрессии является более трудной задачей, и на это есть свои причины.

- Параметры ε и C должны *настраиваться одновременно.*
- По своей сути сама задача регрессии является более сложной, нежели задача классификации.

Принципиальный подход к выбору ε и C остается открытой областью исследований.

И наконец, как и в случае использования машины опорных векторов для распознавания, их применение в задачах нелинейной регрессии может быть реализовано в виде полиномиальной машины обучения, сети на основе радиальных базисных функций или двухслойного персептрона. Ядра скалярных произведений для этих трех методов реализации были представлены в табл. 6.1.

6.9. Резюме и обсуждение

Машины опорных векторов являются элегантным и устоявшимся методом обучения при создании сетей прямого распространения с единственным скрытым слоем нелинейных элементов. Этот метод соответствует принципу минимизации структурного риска, берущему свое начало в теории VC-размерности. Как следует из названия, основная идея создания этой машины состоит в выборе подмножества обучающих данных в качестве опорных векторов. Это подмножество представляет устойчивые свойства всей обучающей выборки. Частными случаями машины опорных векторов являются полиномиальная обучаемая машина, сеть на основе радиальных базисных функций и двухслойный персептрон. Таким образом, несмотря на то, что эти методы реализуют совершенно различные представления встроенных статистических закономерностей, содержащихся в данных обучения, все они происходят от общих корней машины опорных векторов.

В отличие от популярного алгоритма обратного распространения, алгоритм обучения с помощью опорных векторов работает только в пакетном режиме. Существует еще одно важное различие между этими двумя алгоритмами. Алгоритм обратного распространения минимизирует квадратичную функцию потерь, независимо от того, какова задача обучения. В отличие от него, алгоритм нахождения опорных векторов, применяемый для решения задачи распознавания, совершенно отличается от того, который используется в задаче нелинейной регрессии.

- В задаче распознавания алгоритм обучения на основе опорных векторов минимизирует количество обучающих примеров, которые попадают на границу разделения между положительными и отрицательными примерами. Это утверждение является истинным только асимптотически, поскольку вместо функции индикатора $I(\xi_i - 1)$ используются фиктивные переменные ξ_i. Хотя этот критерий в точности и не соответствует минимизации вероятности ошибки классификации, он считается более предпочтительным, чем критерий минимизации среднеквадратической ошибки, на котором основан алгоритм обратного распространения.
- При выполнении задачи нелинейной регрессии алгоритм обучения на основе опорных векторов минимизирует ε-нечувствительную функцию потерь, которая является расширением критерия средней абсолютной ошибки из минимаксной теории. В связи с эти алгоритм является более робастным.

Какой бы ни была задача обучения, машина опорных векторов реализует метод управления сложностью модели, не зависящий от ее размерности. В частности, в пространстве высокой размерности задача сложности модели решается за счет использования "штрафной" гиперплоскости, определенной в (скрытом) пространстве признаков и применяемой в качестве поверхности решения. Результатом становится хорошее качество обобщения. Концентрация внимания на двойственной задаче для решения задачи условной оптимизации приводит к устранению "проклятия размерности". Важной причиной использования двойственной задачи является устранение необходимости определения и вычисления параметров оптимальной гиперплоскости в пространстве данных более высокой размерности.

Обычно обучение машины опорных векторов сводится к задаче квадратичного программирования[5], что привлекательно по двум причинам.

- Процесс обучения гарантированно сходится к глобальному минимуму на поверхности ошибки (ошибкой считается разность между желаемым откликом и фактическим выходом машины опорных векторов).
- Вычисления могут быть реализованы достаточно эффективно.

Более того, при использовании подходящего ядра скалярного произведения машина опорных векторов автоматически вычисляет все важные параметры сети, относящиеся к выбору ядра. Например, в случае сети на основе радиальных базисных функций ядро представляет собой гауссову функцию. Для такого метода реализации количество радиальных базисных функций, их центры, а также линейные веса и значения порогов вычисляются автоматически. В качестве центров радиальных базисных функций выступают опорные векторы, отбираемые согласно стратегии квадратичной оптимизации. Опорные векторы поддержки обычно составляют некоторое подмножество обучающей выборки. Таким образом, создание сети RBF на основе обучения машины опорных векторов можно рассматривать как *частный случай* стратегии строгой интерполяции, описанной в главе 5.

Для решения задачи квадратичного программирования можно использовать некоторые коммерческие библиотеки, предназначенные для решения задач оптимизации[6]. Однако эти библиотеки имеют ограниченное использование. Память, необходимая для решения задачи квадратичного программирования, растет пропорционально квадрату числа примеров обучения. Следовательно, в реальных задачах, включающих обработку нескольких тысяч точек данных, задачу квадратичного программирования

---

[5] В [951] рассматривается алгоритм линейного программирования на основе нормы L_1, а не L_2, которая используется в машинах векторной поддержки. Норма L_1 вектора весов $\mathbf{w}$ определяется как $\|\mathbf{w}\|_1 = \sum_i |w_i|$, где w_i — i-й элемент вектора $\mathbf{w}$. Максимальная граница классификации на основе иормы L_1 смещена в сторону гиперплоскостей, ориентированных по осям, т.е. в сторону векторов весов с малым числом ненулевых элементов.

[6] Среди коммерческих библиотек, предназначенных для решения задач квадратичного программирования, можно выделить следующие: MINOS5.4 [764], LSSOL [358], LOQO [1076], QPOPT и SQOPT [359].

нельзя решить с помощью простого использования коммерческой библиотеки. В [807] предложен новаторский алгоритм декомпозиции, который разбивает задачу оптимизации на последовательность более мелких подзадач. В частности, этот алгоритм декомпозиции учитывает преимущества коэффициентов опорных векторов, которые активны по обе стороны границы классификации, определяемой при $\alpha_i = 0$ и $\alpha_i = C$. В [807] отмечается, что данный алгоритм декомпозиции показал удовлетворительные результаты в приложениях, содержащих до ста тысяч точек данных.

В терминах времени работы машины опорных векторов оказались более медленными по сравнению с другими типами сетей (в том числе по сравнению с многослойными персептронами, обучаемыми по алгоритму обратного распространения). Для такой медлительности имеются две причины.

1. Машина опорных векторов не обеспечивает управления количеством точек данных, выбираемых алгоритмом обучения в качестве опорных векторов.
2. При создании обучаемой машины не учитываются априорные знания о предметной области.

Кратко рассмотрим модификации машин опорных векторов, призванные обойти эти недостатки.

Вопрос о том, как управлять выбором опорных векторов, является крайне сложным, особенно в случае неразделимости классифицируемых множеств и зашумленности данных обучения. В общем случае попытки устранения известных ошибок из данных до начала обучения или удаления их из разложения после обучения не приводят к построению оптимальной гиперплоскости, поскольку для "штрафования" неразделимости нужны ошибки. В [806] исследовалась задача сокращения времени, затрачиваемого машиной опорных векторов на обучение решению задачи классификации. В ней были предложены два новаторских подхода к этой задаче.

- Сама машина опорных векторов использовалась как аппарат нелинейной регрессии для аппроксимации поверхности решений (разделяющей классы) при точности, задаваемой пользователем.
- Процедура обучения машины опорных векторов переформировалась для получения той же точной поверхности решений при меньшем количестве базисных функций.

При первом подходе решение упрощается за счет его аппроксимации линейной комбинацией *подмножества* базисных функций. Полученная машина является естественным расширением машины опорных векторов, создаваемой для аппроксимации функций. Это расширение строится для поиска минимума функционала стоимости

следующего вида:

$$\mathbf{E}(F) = \sum_{i=1}^{N} |d_i - F(\mathbf{x}_i)|_\varepsilon + \frac{1}{2C}\psi(F),$$

где $F(\cdot)$ — аппроксимирующая функция; $\psi(\cdot)$ — некоторый гладкий функционал; $|x|_\varepsilon$ — ε-нечувствительная функция стоимости, определяемая как

$$|x|_\varepsilon = \begin{cases} 0, & \text{если } |x| < \varepsilon, \\ |x| - \varepsilon \text{ в остальных случаях.} \end{cases}$$

Такая ε-нечувствительная функция стоимости обеспечивает робастность решения по отношению к исключениям и нечувствительность к ошибкам, не превышающим некоторого порога ε. Минимум этой функции стоимости $\mathbf{E}(F)$ имеет вид

$$F(\mathbf{x}) = \sum_{i=1}^{N} c_i G(\mathbf{x}, \mathbf{x}_i),$$

где $G(\cdot, \cdot)$ — ядро, зависящее от конкретного выбора гладкой функции $\psi(\cdot)$ и коэффициентов c_i, вычисляемых при решении задачи квадратичного программирования. Решение обычно является *разреженным* (sparse). Это значит, что только небольшое число коэффициентов c_i будет отличаться от нуля, а их количество зависит от параметра ε. При втором подходе прямая задача переформулируется так, что она имеет ту же начальную структуру, что и исходная, но с одним отличием: в формулировку включается ядро скалярного произведения $K(\mathbf{x}, \mathbf{x}')$. Оба подхода призваны уменьшить сложность машины опорных векторов в задачах нелинейной регрессии.

В заключение, возвращаясь к вопросу априорных знаний, заметим, что производительность обучаемой машины может быть существенно увеличена включением этих знаний в архитектуру машины [4]. В литературе рассматриваются два способа использования априорных знаний.

- В качестве дополнительного слагаемого в функции стоимости. Тогда обучаемая машина будет строить функцию с учетом априорных знаний. Именно это происходит при использовании регуляризации.
- В качестве виртуальных примеров, генерируемых на основе обучающего множества. Тогда обучаемая машина сможет легко извлекать априорные знания из искусственно расширенного множества примеров.

При втором подходе процесс обучения может сильно замедлиться в связи с корреляцией искусственных данных и увеличением размера обучающего множества. Однако преимущество второго подхода состоит в том, что его легче реализовать для всех

типов априорных знаний и обучаемых машин. Один из способов реализации второго подхода предложен ниже [946].

1. Машина опорных векторов обучается обычным образом на предложенном множестве данных с целью извлечения опорных векторов.
2. Путем применения априорных знаний в форме желаемых инвариантных преобразований к векторам, полученным в п. 1, генерируются искусственные примеры, называемые *виртуальными опорными векторами* (virtual support vector).
3. На искусственно расширенном множестве примеров обучается другая машина опорных векторов.

Этот метод позволяет повысить точность классификации за счет увеличения времени обучения, поскольку он требует двух циклов обучения. Однако в этом случае правила классификации строятся на основе большего количества опорных векторов.

Задачи

Оптимальная разделяющая гиперплоскость

6.1. Рассмотрим гиперплоскость для линейного разделения образов, определяемую уравнением

$$\mathbf{w}^T\mathbf{x} + b = 0,$$

где $\mathbf{w}$ — вектор весов; b — порог; $\mathbf{x}$ — входной вектор. Гиперплоскость называется соответствующей *канонической паре* (canonical pair) $(\mathbf{w}, b)$, если для множества входных образов $\{\mathbf{x}_i\}_{i=1}^N$ выполняется дополнительное условие

$$\min_{i=1,2,\ldots,N} |\mathbf{w}^T\mathbf{x}_i + b| = 1.$$

Покажите, что при выполнении этого условия ширина границы разделения между двумя классами равна $2/||\mathbf{w}||$.

6.2. В контексте неразделимых образов докажите следующее утверждение. Наличие ошибок классификации означает неразделимость множеств. Обратное утверждение *не* всегда верно.

6.3. Для прямой задачи оптимизации разделяющей гиперплоскости для неразделимых множеств сформулируйте двойственную задачу, как это сделано в разделе 6.3.

6.4. Для оценки ожидаемой ошибки тестирования, генерируемой оптимальной гиперплоскостью в случае неразделимых множеств, воспользуйтесь методом, описанным в главе 4. Рассмотрите различные ситуации, которые могут возникнуть при использовании этого метода, если один из примеров удаляется из обучающего множества и решение строится на основе оставшихся.

6.5. Положение оптимальной гиперплоскости в пространстве данных определяется точками данных, выбранных в качестве опорных векторов. Если данные зашумлены, возникает вопрос о робастности границ разделения к наличию шума. При внимательном изучении оптимальной гиперплоскости робастность границы разделения по отношению к шуму подтверждается. Дайте этому разумное объяснение.

Ядро скалярного произведения

6.6. Ядро скалярного произведения $K(\mathbf{x}_i, \mathbf{x}_j)$, построенное на множестве примеров обучения $\mathbf{T}$ размера N, образует матрицу размерности $N \times N$:

$$\mathbf{K} = \{K_{ij}\}_{(i,j)=1}^{N},$$

где $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Матрица $\mathbf{K}$ является положительной, если положительны все ее элементы. Используя преобразования подобия

$$\mathbf{K} = \mathbf{Q}\boldsymbol{\lambda}\mathbf{Q}^T,$$

где $\boldsymbol{\lambda}$ — диагональная матрица, состоящая из собственных значений; $\mathbf{Q}$ — матрица, составленная из соответствующих собственных векторов, сформулируйте выражение для ядра скалярного произведения $K(\mathbf{x}_i, \mathbf{x}_j)$ в терминах собственных значений и собственных векторов матрицы $\mathbf{K}$. Какие выводы можно сделать из этого представления?

6.7. а) Докажите *свойство унитарной инвариантности* (unitary invariance property) ядра скалярного произведения $K(\mathbf{x}, \mathbf{x}_i)$

$$K(\mathbf{x}, \mathbf{x}_i) = K(\mathbf{Q}\mathbf{x}, \mathbf{Q}\mathbf{x}_i),$$

где $\mathbf{Q}$ — унитарная матрица, т.е.

$$\mathbf{Q}^{-1} = \mathbf{Q}^T.$$

б) Покажите, что все три ядра скалярного произведения, приведенные в табл. 6.1, удовлетворяют этому свойству.

6.8. Ядро скалярного произведения двухслойного персептрона определяется следующим образом:

$$K(\mathbf{x}, \mathbf{x}_i) = \tanh(\beta_0 \mathbf{x}^T \mathbf{x}_i + \beta_1).$$

Найдите несколько значений констант β_0 и β_1, для которых теорема Мерсера не удовлетворяется.

Классификация множеств

6.9. Для решения задачи XOR в полиномиальной обучаемой машине используется ядро скалярного произведения вида

$$K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}^T \mathbf{x}_i)^p.$$

Каково минимальное значение степени p, при которой решается задача XOR? Предполагается, что p — целое положительное число. Что произойдет в случае использования значения p, превышающего минимальное?

6.10. На рис. 6.9 показана функция XOR, определенная на трехмерном множестве входных сигналов $\mathbf{x}$:

$$XOR(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \mathbf{x}_3,$$

где символ$\oplus$ обозначает логический оператор "исключающего ИЛИ". Создайте полиномиальную обучаемую машину, распознающую два класса точек, определяемых выходом этого оператора.

6.11. На протяжении всей главы обсуждался вопрос использования машины опорных векторов для задач двоичной классификации. Проанализируйте, как можно использовать машину опорных векторов для решения задачи классификации порядка M, где $M > 2$.

Нелинейная регрессия

6.12. Двойственная задача, сформулированная в разделе 6.8, для случая использования машины опорных векторов для решения задачи нелинейной регрессии имеет следующее ограничение:

$$\sum_{i=1}^{N} (\alpha_i - \alpha_i') = 0,$$

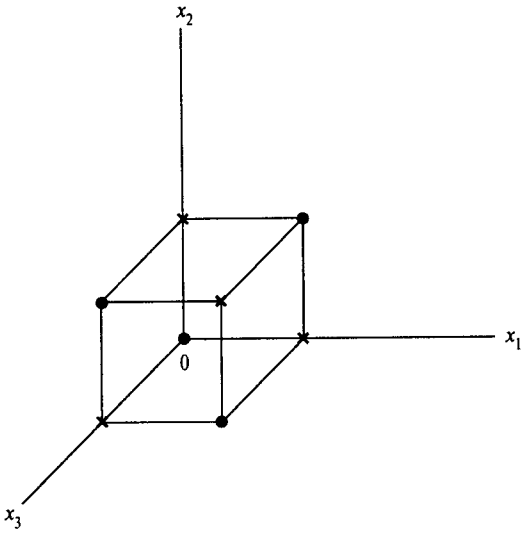


Рис. 6.9. Оператор "исключающего ИЛИ" в трехмерном пространстве

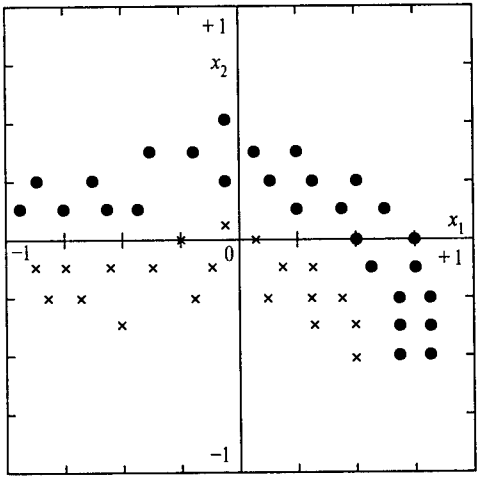


Рис. 6.10. Графическое представление классов

где α_i и α_i' — множители Лагранжа. Покажите, что это ограничение является следствием минимизации Лагранжиана относительно порога b, т.е. первого элемента w_0 вектора весов $\mathbf{w}$, соответствующего $\varphi(\mathbf{x}) = 1$.

Преимущества и недостатки

6.13. а) Проанализируйте преимущества и недостатки машин опорных векторов по сравнению с сетями на основе радиальных базисных функций (RBF) при решении следующих задач: классификации образов (1) и нелинейной регрессии (2).

б) Выполните сравнительный анализ машины опорных векторов с многослойным персептроном, обучаемым по алгоритму обратного распространения.

Компьютерное моделирование

6.14. На рис. 6.10 показано множество точек, соответствующих двум классам — $\mathbf{C}_1$ и $\mathbf{C}_2$. Координаты x_1 и x_2 изменяются на интервале от -1 до $+1$. Используя ядро скалярного произведения

$$K(\mathbf{x}, \mathbf{t}) = \exp(-||\mathbf{x} - \mathbf{t}||^2),$$

постройте оптимальную разделяющую гиперплоскость для этого набора данных.

6.15. Компьютерный эксперимент, описанный в разделе 6.6, был проведен для классификации двух пересекающихся гауссовых распределений. В этом эксперименте использовался параметр регуляризации $C = 0,1$. Ширина радиальных базисных функций, использованных для построения скалярного ядра, составляла $\sigma^2 = 4$. Повторите этот же эксперимент, используя другие параметры регуляризации:

а) $C = 0,05$;

б) $C = 0,2$.

Прокомментируйте полученный результат в свете значений, полученных в разделе 6.6.

6.16. При использовании сетей на основе радиальных базисных функций для решения задач нелинейной регрессии часто оказывается, что применение нелокализованных базисных функций, таких как мультиквадратичная, обеспечивает более высокую точность решения, чем локализованных, таких как функция Гаусса. Можно высказать предположение, что аналогичная ситуация характерна и для машин опорных векторов, так как использование (неограниченных) полиномиальных обучаемых машин обеспечивает более высокую точность, чем (ограниченных) машин на основе радиальных базисных функций. Проверьте правильность этого предположения экспериментально для задачи классификации.

Глава 7

Ассоциативные машины

7.1. Введение

В предыдущих трех главах описывались три различных подхода к обучению с учителем. В главе 4 описывался многослойный персептрон, обучаемый по методу обратного распространения и реализующий одну из форм глобальной оптимизации (во всем пространстве весовых коэффициентов). Сети на основе радиальных базисных функций, описанные в главе 5, благодаря своей структуре обеспечивают локальную оптимизацию. Машины опорных векторов, рассмотренные в главе 6, базируются на теории VC-измерений. В этой главе мы обсудим еще один класс методов, предназначенных для решения задач обучения с учителем. Представленный здесь подход основан на общеизвестном принципе "разделяй и властвуй" (divide and conquer).

В соответствии с этим принципом сложные вычислительные задачи решаются при помощи их разбиения на множество небольших и простых задач с последующим объединением полученных решений. При обучении с учителем вычислительная простота достигается за счет распределения задачи обучения среди множества *экспертов*, которые, в свою очередь, разбивают входное пространство на множество подпространств. Комбинацию таких экспертов и называют *ассоциативной машиной* (committee machine). По сути она интегрирует знания, накопленные экспертами, в общее решение, которое имеет приоритет над каждым решением отдельного эксперта. Идея ассоциативной машины появилась еще в 1965 году в [786]. Предложенная в этой работе сеть состоит из слоя элементарных персептронов, за которым следует второй слой — принятия решения.

Ассоциативные машины являются универсальными аппроксиматорами. Их можно разбить на две основные категории.

1. *Статические структуры* (static structure). В этом классе ассоциативных машин отклики различных предикторов (экспертов) объединяются с помощью некоторого механизма, не учитывающего входной сигнал. Поэтому они и получили название "статические". Эта категория структур работает на основе следующих методов.
   - *Усреднение по ансамблю* (ensemble averaging). Выходной сигнал вычисляется

- *Усреднение по ансамблю* (ensemble averaging). Выходной сигнал вычисляется как линейная комбинация выходов отдельных предикторов.
- *Усиление* (boosting), при котором слабый алгоритм обучения превращается в алгоритм, достигающий произвольной заданной точности.

2. *Динамические структуры* (dynamic structure). В этом втором классе ассоциативной машины входной сигнал непосредственно учитывается в механизме объединения выходных сигналов экспертов (благодаря этому свойству данные машины и получили название "динамических"). Можно выделить две различные реализации динамических структур.
   - *Смешение мнений экспертов* (mixture of experts), при котором отклики отдельных экспертов нелинейно объединяются в единую шлюзовую сеть (gating network).
   - *Иерархическое объединение мнений экспертов*, при котором отклики отдельных экспертов нелинейно объединяются с помощью нескольких шлюзовых сетей, организованных в иерархическую структуру.

   При смешении мнений экспертов принцип "разделяй и властвуй" применяется всего один раз, в то время как при иерархическом смешении он применяется неоднократно для каждого слоя иерархии.

Смешение мнений экспертов и иерархическое смешение можно рассматривать как примеры *модульных* (modular) сетей. Формальное определение *модульности* было приведено в [804]. Оно звучит следующим образом.

> *Нейронная сеть является модульной, если выполняемые ею вычисления можно распределить по нескольким подсистемам, которые обрабатывают различные входные сигналы и не пересекаются в своей работе друг с другом. Выходные сигналы этих подсистем объединяются модулем интеграции, выход которого не имеет обратной связи с подсистемами. В частности, модуль интеграции принимает решение о том, как выходные сигналы подсистем объединяются в общий выходной сигнал системы, и определяет, на каких примерах следует обучать конкретные модули.*

Это определение модульности исключает из рассмотрения класс статических ассоциативных машин, так как в последних не существует никаких элементов интегрирования, которые выполняют принятие решений.

Структура главы

Эту главу формально можно разбить на две части. В первой части (разделы 7.2–7.5) рассматривается класс статических структур. В частности, в разделе 7.2 описывается метод усреднения по ансамблю, компьютерное моделирование которого будет выполнено в разделе 7.3. В разделе 7.4 мы рассмотрим технику усиления, а в разделе 7.5 приведем результаты ее компьютерного моделирования.

Во второй части главы (разделы 7.6–7.13) рассматриваются динамические структуры. В частности, в разделе 7.6 представлена структура смешения мнений экспертов, или ME-структура (mixture of experts), как ассоциативная гауссова модель смешения (associative Gaussian mixture model). В разделе 7.7 описывается более общий случай – иерархическое смешение мнений экспертов, или HME-структура (hierarchical mixture of experts). Эта вторая модель близка к стандартным деревьям решений. В разделе 7.8 речь пойдет о том, как стандартное дерево решений можно использовать для решения задачи выбора модели (т.е. количества экспертных и шлюзовых сетей) в HME. В разделе 7.9 будут определены апостериорные вероятности, используемые при описании алгоритмов обучения HME. В разделе 7.10 будут заложены основы для решения задачи оценки параметров и построена функция правдоподобия для модели HME. В разделе 7.11 представлен обзор стратегий обучения, а в разделе 7.12 последует детальное описание так называемого *алгоритма EM*, применению которого для модели HME посвящен раздел 7.13.

Как всегда, глава завершится заключительными выводами и обсуждением.

7.2. Усреднение по ансамблю

На рис. 7.1 показано множество отдельно обучаемых нейронных сетей (экспертов) с общим входным сигналом. Их выходные сигналы некоторым образом комбинируются, формируя общий выход системы y. Для того чтобы упростить выкладки, предположим, что выходы экспертов представляют собой скалярные величины. Представленный здесь подход носит название *метода усреднения по ансамблю* (ensemble averaging method)[1]. Использование этого метода обусловлено двумя основными причинами.

- Если множество экспертов, показанное на рис. 7.1, заменить единой нейронной сетью, получится сеть, содержащая гораздо большее количество настраиваемых параметров. Естественно, время обучения такой сети будет существенно больше времени параллельного обучения множества экспертов.
- Риск избыточного обучения (overfitting) возрастает, если количество настраиваемых параметров существенно больше размера множества данных обучения.

При использовании ассоциативных машин (см. рис. 7.1) предполагается, что обучаемые по отдельности эксперты будут сходиться к разным локальным минимумам поверхности ошибок, в результате чего некоторая комбинация их выходных сигналов приведет к повышению эффективности сети.

---

[1] Методы усреднения по множеству обсуждаются в [828], в которой собрана довольно большая библиография по этой теме. К другим рекомендуемым работам относятся [424] и [1164].

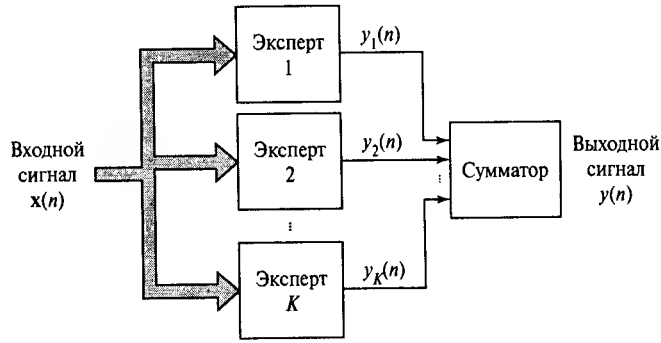


Рис. 7.1. Блочная диаграмма ассоциативной машины, основанной на усреднении по ансамблю

В первую очередь рассмотрим единую нейронную сеть, обучаемую на некотором множестве примеров. Пусть $\mathbf{x}$ — некоторый ранее не встречавшийся входной вектор; d — соответствующий ему желаемый отклик (метка класса или численный отклик). Эти величины представляют собой реализацию случайного вектора $\mathbf{X}$ и случайной переменной D. Пусть $F(\mathbf{x})$ — реализуемая сетью функция отображения входного сигнала в выходной. Тогда в свете дилеммы смещения и дисперсии, описанной в главе 2, среднеквадратическую ошибку между функцией $F(\mathbf{x})$ и условным математическим ожиданием $E(D|\mathbf{X} = \mathbf{x})$ можно разложить на слагаемые смещения и дисперсии

$$E_{\mathbf{D}}[(F(\mathbf{x}) - E[D|\mathbf{X} = \mathbf{x}])^2] = B_{\mathbf{D}}(F(\mathbf{x})) + V_{\mathbf{D}}(F(\mathbf{x})), \tag{7.1}$$

где $B_{\mathbf{D}}(F(\mathbf{x}))$ — квадрат смещения

$$B_{\mathbf{D}}(F(\mathbf{x})) = (E_{\mathbf{D}}[F(\mathbf{x})] - E[D|\mathbf{X} = \mathbf{x}])^2, \tag{7.2}$$

а $V_{\mathbf{D}}(F(\mathbf{x}))$ — дисперсия

$$V_{\mathbf{D}}(F(\mathbf{x})) = E_{\mathbf{D}}[(F(\mathbf{x}) - E_{\mathbf{D}}[F(\mathbf{x})])^2]. \tag{7.3}$$

Математическое ожидание $E_{\mathbf{D}}$ вычисляется в пространстве $\mathbf{D}$, которое определяется как *пространство, охватывающее распределение всех обучающих множеств (т.е. входных сигналов и целевых выходов) и распределение всех начальных условий.*

Существуют различные способы отдельного обучения различных экспертных сетей, а также различные методы объединения их выходных сигналов. В данной главе будет рассмотрена ситуация, в которой экспертные сети имеют одинаковую конфигурацию, но начинают обучение из различных исходных состояний. Для объединения их входных сигналов будет использоваться простой *блок усреднения по ансамблю*[2].

[2] Использование усреднения по ансамблю при построении ассоциативной машины на множестве различных начальных условий ранее предлагалось многими практиками нейронных сетей. Однако статистический анализ, представленный в [770], и процедура обучения ассоциативных машин, определенная для метода усред-

Пусть $\mathbf{\Phi}$ — *пространство всех исходных состояний*; $F_I(\mathbf{x})$ — среднее всех функций отображения входного сигнала в выходной в экспертных сетях на множестве исходных состояний. По аналогии с формулой (7.1) можно записать следующее:

$$E_{\mathbf{\Phi}}[(F_I(\mathbf{X}) - E[D|\mathbf{X} = \mathbf{x}])^2] = B_{\mathbf{\Phi}}(F(\mathbf{x})) + V_{\mathbf{\Phi}}(F(\mathbf{x})), \tag{7.4}$$

где $B_{\mathbf{\Phi}}(F(\mathbf{x}))$ — квадрат смещения, определенного в пространстве $\mathbf{\Phi}$:

$$B_{\mathbf{\Phi}}(F(\mathbf{x})) = (E_{\mathbf{\Phi}}[F_I(\mathbf{x})] - E[D|\mathbf{X} = \mathbf{x}])^2, \tag{7.5}$$

а $V_{\mathbf{\Phi}}(F(\mathbf{x}))$ — соответствующая дисперсия:

$$V_{\mathbf{\Phi}}(F(\mathbf{x})) = E_{\mathbf{\Phi}}[(F_I(\mathbf{x}) - E_{\mathbf{\Phi}}[F(\mathbf{x})])^2]. \tag{7.6}$$

Математическое ожидание $E_{\mathbf{\Phi}}$ вычисляется по всему пространству $\mathbf{\Phi}$.

Из определения пространства $\mathbf{D}$ видно, что оно представляет собой произведение пространства исходных условий $\mathbf{\Phi}$ на *оставшееся пространство* (remnant space) $\mathbf{D}'$. Следовательно, по аналогии с (7.1) можно записать:

$$E_{\mathbf{D}'}[(F_I(\mathbf{x}) - E[D|\mathbf{X} = \mathbf{x}])^2] = B_{\mathbf{D}'}(F_I(\mathbf{x})) + V_{\mathbf{D}'}(F_I(\mathbf{x})), \tag{7.7}$$

где $B_{\mathbf{D}'}(F(\mathbf{x}))$ — квадрат смещения, определенного в пространстве $\mathbf{\Phi}'$:

$$B_{\mathbf{D}'}(F_I(\mathbf{x})) = (E_{\mathbf{D}'}[F_I(\mathbf{x})] - E[D|\mathbf{X} = \mathbf{x}])^2, \tag{7.8}$$

а $V_{\mathbf{D}'}(F(\mathbf{x}))$ — соответствующая дисперсия:

$$V_{\mathbf{D}'}(F_I(\mathbf{x})) = E_{\mathbf{D}'}[(F_I(\mathbf{x}) - E_{\mathbf{D}'}[F_I(\mathbf{x})])^2]. \tag{7.9}$$

Из определений пространств $\mathbf{D}$, $\mathbf{\Phi}$ и $\mathbf{D}'$ видно, что

$$E_{\mathbf{D}'}[F_I(\mathbf{x})] = E_{\mathbf{D}}[F_I(\mathbf{x})]. \tag{7.10}$$

нения по множеству начальных условий, явились первыми исследованиями такого рода. Представленные в этой работе экспериментальные результаты подтвердили значительное уменьшение дисперсии при усреднении по пространству начальных условий.

Согласно [770], в ассоциативных машинах, использующих усреднение по пространству начальных состояний, *не рекомендуется* применять популярные модификации алгоритма обучения, типа уменьшения весов или раннего останова.

Отсюда следует, что выражение (7.8) можно переписать в эквивалентной форме:

$$B_{\mathbf{D}'}(F_I(\mathbf{x})) = (E_{\mathbf{D}}[F(\mathbf{x})] - E[D|\mathbf{X} = \mathbf{x}])^2 = B_{\mathbf{D}}(F(\mathbf{x})). \tag{7.11}$$

Далее рассмотрим дисперсию $V_{\mathbf{D}'}(F(\mathbf{x}))$ в выражении (7.9). Так как дисперсия случайной переменной равна среднеквадратическому значению этой переменной за вычетом квадрата смещения, то можно записать:

$$V_{\mathbf{D}'}(F_I(\mathbf{x})) = E_{\mathbf{D}'}[(F_I(\mathbf{x}))^2] - (E_{\mathbf{D}'}[F_I(\mathbf{x})])^2 = E_{\mathbf{D}'}[(F_I(\mathbf{x}))^2] - (E_{\mathbf{D}}[F_I(\mathbf{x})])^2. \tag{7.12}$$

В последнем равенстве используется соотношение (7.10). Аналогично, выражение (7.3) можно переписать в следующей эквивалентной форме:

$$V_{\mathbf{D}}(F_I(\mathbf{x})) = E_{\mathbf{D}}[(F(\mathbf{x}))^2] - (E_{\mathbf{D}}[F(\mathbf{x})])^2. \tag{7.13}$$

Заметим, что среднеквадратическое значение функции $F(\mathbf{x})$ во всем пространстве $\mathbf{D}$ должно быть не меньше значения среднеквадратической функции $F_I(\mathbf{x})$ в пространстве дополнения $\mathbf{D}'$, т.е.

$$E_{\mathbf{D}}[F(\mathbf{x})^2] \geq E_{\mathbf{D}'}[(F_I(\mathbf{x}))^2].$$

В свете этого неравенства сравнение (7.12) и (7.13) приводит к следующему заключению:

$$V_{\mathbf{D}'}(F_I(\mathbf{x})) \leq V_{\mathbf{D}}(F(\mathbf{x})). \tag{7.14}$$

Таким образом, из выражений (7.11) и (7.14) можно сделать следующие два вывода.

1. Смещение усредненной по ансамблю функции $F_I(\mathbf{x})$ для ассоциативной машины, показанной на рис. 7.1, имеет то же значение, что и для функции $F(\mathbf{x})$ отдельной нейронной сети.
2. Дисперсия усредненной по ансамблю функции $F_I(\mathbf{x})$ не меньше дисперсии отдельных функций $F(\mathbf{x})$.

Эти теоретические рассуждения определяют стратегию обучения, которая приводит к уменьшению общей ошибки ассоциативной машины за счет *варьирования начальных состояний* [770]: отдельные "эксперты" машины целенаправленно *обучаются с избытком* (overtrained). Этому есть следующее объяснение: поскольку мы имеем дело с отдельными экспертами, смещение уменьшается за счет дисперсии. Далее дисперсия уменьшается путем усреднения параметров экспертов ансамбля по начальным условиям при неизменном смещении.

7.3. Компьютерный эксперимент 1

В этом компьютерном эксперименте, посвященном исследованию метода *усреднения по ансамблю*, снова рассмотрим задачу бинарной классификации, с которой мы уже сталкивались в предыдущих трех главах. Это задача классификации двух классов, определяемых двумерными пересекающимися гауссовыми распределениями. Эти два распределения имеют различные средние значения и дисперсии. Класс $\mathbf{C}_1$ имеет такие статистические характеристики:

$$\boldsymbol{\mu}_1 = [0, 0]^T,$$
$$\sigma_1^2 = 1,$$

а класс $\mathbf{C}_2$ — следующие:

$$\boldsymbol{\mu}_2 = [2, 0]^T,$$
$$\sigma_2^2 = 4.$$

Графики этих двух распределений представлены на рис. 4.13.

Предполагается, что эти два класса равновероятны. Стоимость ошибки классификации предполагается равной в обоих классах, а стоимость корректной классификации предполагается равной нулю. При таких условиях (оптимальный) классификатор Байеса обеспечивает вероятность корректной классификации, составляющую p_c =81,51%. Обоснование этого результата приводится в главе 4.

В компьютерном эксперименте, описанном в главе 4, для персептрона с двумя скрытыми элементами, обучаемого с помощью алгоритма обратного распространения, удалось обеспечить вероятность корректной классификации порядка 80%. В настоящем разделе для решения задачи будем использовать ассоциативную машину, состоящую из десяти экспертов, при этом каждый из экспертов представляет собой многослойный персептрон с двумя скрытыми нейронами.

Все эксперты обучаются отдельно с помощью алгоритма обратного распространения со следующими параметрами.

Параметр скорости обучения η =0,1.

Константа фактора момента α =0,5.

Обучающее множество состоит из 500 образов. Все эксперты обучаются на одном и том же множестве примеров, но имеют различные исходные состояния. В частности, исходные значения синаптических весов и порогов выбираются случайным образом с помощью генератора случайных чисел с равномерным распределением в диапазоне $[-1; 1]$.

ТАБЛИЦА 7.1. Эффективность классификации отдельных экспертов в ассоциативной машине

Эксперт	*Процент корректной классификации*
Сеть 1	80,65
Сеть 2	76,91
Сеть 3	80,06
Сеть 4	80,47
Сеть 5	80,44
Сеть 6	76,89
Сеть 7	80,55
Сеть 8	80,47
Сеть 9	76,91
Сеть 10	80,38

В табл. 7.1 представлены результаты классификации отдельных экспертов, обучаемых на данном множестве образов. Вероятность корректной классификации была получена путем вычисления среднего арифметического по множеству результатов в табл. 7.1 и составила $p_{c,\text{av}}$ =79,37%. С другой стороны, используя метод *усреднения по ансамблю*, т.е. просто суммируя результаты всех 10 экспертов и только затем вычисляя вероятность корректной классификации, получим результат $p_{c,\text{ens}} = 80,27\%$. Налицо улучшение результата на 0,9%. Более высокое значение $p_{c,\text{ens}}$ по сравнению с $p_{c,\text{av}}$ наблюдалось во всех попытках данного эксперимента. Результаты классификации каждый раз вычислялись на тестовом множестве из 32000 примеров.

Анализируя результаты эксперимента, можно утверждать следующее. Эффективность классификации улучшается за счет переобучения отдельных персептронов (экспертов) и суммирования их выходных сигналов в единый выходной сигнал ассоциативной машины, на основании которого уже и принимается решение.

7.4. Метод усиления

Как уже говорилось во введении, метод усиления является еще одним способом реализации класса "статических" ассоциативных машин. В ассоциативных машинах, основанных на усреднении по ансамблю, все эксперты обучаются на одном и том же множестве данных. Сети отличаются друг от друга только выбором исходного состояния. В противоположность этому сети-эксперты, работающие на основе метода усиления, обучаются на примерах, принадлежащих совершенно различным распределениям. Это самый общий метод из тех, которые можно использовать для улучшения производительности *любого* алгоритма обучения.

Метод усиления[3] (boosting) может быть реализован тремя способами.

1. *Усиление за счет фильтрации* (boosting by filtering). Этот подход предполагает отбор (фильтрацию) примеров обучения различными версиями слабого алгоритма обучения. При этом предполагается доступность большого (в идеале — бесконечного) множества примеров. Во время обучения примеры могут быть отбракованы или сохранены. Преимуществом этого подхода является то, что по сравнению с двумя остальными он не предъявляет больших требований к памяти.
2. *Усиление за счет формирования подвыборок* (boosting by subsampling). Этот подход предполагает наличие множества примеров обучения фиксированного размера. Подвыборки составляются во время обучения в соответствии с заданным распределением вероятности. Ошибка вычисляется относительно фиксированного множества примеров обучения.
3. *Усиление путем перевзвешивания* (boosting by reweighting). Третий подход связан с обработкой фиксированного множества примеров. При этом предполагается, что слабый алгоритм обучения может получать "взвешенные" примеры. Ошибка вычисляется относительно взвешенных примеров.

В этом разделе описываются два алгоритма усиления. Первый из них, согласно [940], относится к первому вышеописанному подходу. Второй алгоритм, называемый AdaBoost [319], [320], относится ко второму подходу.

Усиление за счет фильтрации

Исходная идея усиления, описанная в [940], берет свое начало в *независимой от распределения* (distribution-free) или *статистически аппроксимативно корректной* (probably approximately correct — PAC) модели обучения. Из обсуждения модели PAC в главе 2 ясно, что *понятие* или *концепт* (concept) можно рассматривать как булеву функцию в предметной области экземпляров, которая включает коды всех интересующих нас объектов. При обучении PAC-машина пытается идентифицировать неизвестный двоичный концепт на основе случайно выбранных его экземпляров. Более строго, целью обучаемой машины является поиск гипотезы или правила прогнозирования с вероятностью ошибки, не превышающей некоторой небольшой наперед заданной величины ϵ, причем это условие должно выполняться для всех входных распределений. Именно по этой причине обучение PAC называют также *сильной моделью обучения* (strong learning model). Так как примеры имеют случайную природу, обучаемая машина, вероятнее всего, не сможет составить представление о неизвестном понятии, если множество примеров будет не представительным. Поэтому модель обучения должна

[3] Основными работами по теории усиления и связанными с ней экспериментальными приложениями являются следующие: [940], [267], [266], [317], [153], [319], [320], [318], [939] и [941]. Они перечислены примерно в хронологическом порядке Лучшими ссылками по трем основным подходам к усилению являются [940] (фильтрация), [319] (подвыборка (resampling)) и [317] (перевзвешивание).

находить хорошую аппроксимацию неизвестного понятия с вероятностью $(1-\delta)$, где δ — некоторое малое положительное число.

В вариации PAC-обучения, называемой *слабой моделью обучения* (weak learning model), требования к поиску неизвестного понятия сильно ослаблены. Обучаемая машина должна построить гипотезу с вероятностью ошибки, несколько меньшей значения 1/2. При случайном "угадывании" двоичной разметки для каждого примера гипотеза с равной вероятностью может оказаться как правильной, так и неверной. Значит, в этом случае вероятность ошибки составляет 1/2. Отсюда следует, что для слабой модели обучения достаточно достичь уровня эффективности, несколько превосходящего абсолютно случайный выбор. Понятие слабой обучаемости было введено в [550], где была сформулирована *задача усиления гипотезы* (hypothesis boosting problem), которая сводится к следующему вопросу.

Эквивалентны ли понятия сильного и слабого обучения?

Другими словами, является ли любой слабо обучаемый класс понятий также и сильно обучаемым? Положительный ответ на этот вопрос, несколько удивительный на первый взгляд, дан в [940], где представлено конструктивное доказательство эквивалентности слабого и сильного обучения. В частности, в [940] был предложен алгоритм, преобразующий слабую модель обучения в сильную. Это достигается благодаря изменению распределения примеров обучения таким образом, чтобы сильная модель строилась "вокруг" слабой.

При использовании усиления за счет фильтрации ассоциативная машина состоит из трех экспертов или подгипотез. Алгоритм, используемый для такого обучения, называют алгоритмом *усиления* (boosting algorithm). При этом три эксперта произвольно маркируются как "первый", "второй" и "третий". Они обучаются по отдельности следующим образом.

1. Первый эксперт обучается на множестве, состоящем из N_1 примеров.
2. Обученный первый эксперт используется для *фильтрации* (filter) второго множества примеров следующим образом.
   - Случайный выбор моделируется подбрасыванием монетки.
   - Если выпадает *решка* (head), новый пример "пропускается" через первого эксперта и корректно классифицированные примеры отклоняются до тех пор, пока не возникнет ошибка классификации. Пример, приведший к ошибке классификации, добавляется в множество примеров для обучения второго эксперта.
   - Если выпадает *орел* (tail), производятся действия, прямо противоположные вышеописанным, т.е. примеры "пропускаются" через первого эксперта и отклоняются до тех пор, пока очередной пример не будет классифицирован правильно. Этот корректно классифицированный пример добавляется в множество примеров, подготавливаемых для обучения второго эксперта.

- Этот процесс продолжается до тех пор, пока все множество из N_1 примеров не будет отфильтровано первым экспертом. Отфильтрованное таким образом множество примеров подается для обучения второго эксперта.

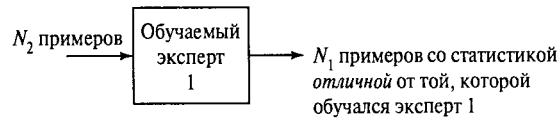
Процедура подбрасывания монетки гарантирует, что при тестировании первого эксперта на втором наборе примеров ошибка классификации составит 1/2. Другими словами, второе множество из N_1 примеров, доступное для обучения второго эксперта, имеет распределение, полностью отличное от распределения первого множества из N_1 примеров, использованных ранее для обучения первого эксперта. Таким образом, второй эксперт вынужден учиться на распределении, отличающемся от использованного для обучения первого эксперта.

3. После обучения второго эксперта множество примеров обучения для третьего эксперта формируется следующим образом.
   - Новый пример "пропускается" через первого и второго экспертов. Если решения обоих экспертов совпадают, пример отклоняется; если они расходятся в своих мнениях, данный пример включается в множество примеров обучения третьего эксперта.
   - Этот процесс продолжается до тех пор, пока не будет отфильтровано все множество из N_1 примеров. Полученное множество примеров затем используется для обучения третьего эксперта.

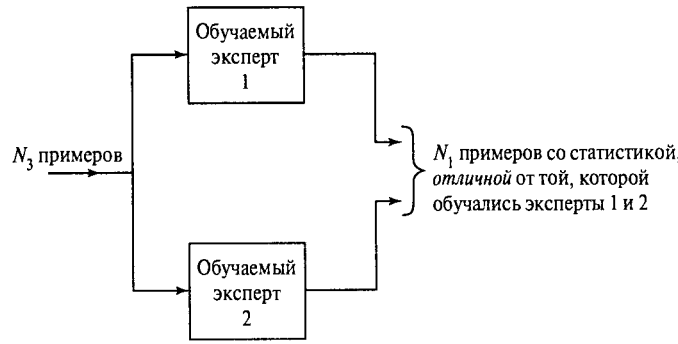
   После окончания обучения третьего эксперта на отфильтрованном множестве примеров процесс обучения всей ассоциативной машины считается завершенным.

Описанная выше процедура фильтрации графически представлена на рис. 7.2.

Пусть N_2 — количество примеров, отфильтрованных первым экспертом для получения обучающего множества второго эксперта, содержащего N_1 элементов. Заметим, что число N_1 фиксировано, а число N_2 зависит от ошибки обобщения первого эксперта. Пусть N_3 — количество примеров, обработанных в процессе совместной фильтрации первым и вторым экспертами для формирования обучающего множества третьего эксперта из N_1 элементов. Так как это же число (N_1) ранее использовалось для обучения первого эксперта, общий объем данных, необходимый для обучения ассоциативной машины, составляет $N_4 = N_1 + N_2 + N_3$. Однако при этом вычислительная стоимость обучения зависит от $3*N_1$ примеров, так как N_1 — это количество примеров, действительно используемых для обучения всех трех экспертов. Таким образом, можно утверждать, что описанный здесь алгоритм усиления является действительно "интеллектуальным" (smart): для работы ассоциативной машины требуется большой объем примеров, но на самом деле для реального обучения будет использоваться только небольшое подмножество этих данных.



а) Фильтрация примеров, выполненная экспертом 1



б) Фильтрация примеров, выполненная экспертами 2 и 3

Рис. 7.2. Графически представленный процесс усиления за счет фильтрации

Еще одним вопросом, на который хотелось бы обратить внимание, является следующий. Операция фильтрации, выполняемая первым экспертом, и операция совместной фильтрации, совместно выполняемая первым и вторым экспертами, заставляют второго и третьего экспертов сосредоточиться на "тяжелых для усвоения" частях распределения.

В теоретическом выводе алгоритма усиления, впервые представленном в [940], для оценки производительности ассоциативной машины на не встречавшихся ранее примерах использовалось простое голосование. А именно, на вход ассоциативной машины подавался тестовый пример. Если решения первого и второго экспертов совпадали, для маркировки использовался именно этот класс. В противном случае использовалась оценка третьего эксперта. Однако экспериментально [266], [267] было определено, что *сложение* соответствующих выходов трех экспертов приводит к лучшей производительности, нежели голосование. Например, в задаче оптического распознавания символов (optical character recognition — OCR) операция сложения выполнялась сложением выходов "цифра 0" всех трех экспертов; аналогично — для всех остальных девяти цифр.

Предположим, что все три эксперта (т.е. подгипотезы) имеют уровень ошибок $\epsilon < 1/2$ по отношению к распределениям, на которых происходило их обучение. Это значит, что все три модели обучения являются слабыми. В [940] было доказано, что общий уровень ошибок ассоциативной машины в данном случае ограничен следующей величиной:

$$g(\epsilon) = 3\epsilon^2 - 2\epsilon^3. \tag{7.15}$$

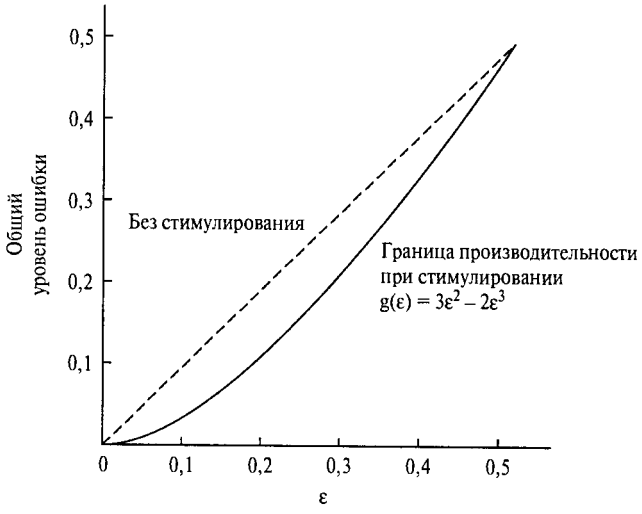


Рис. 7.3. График зависимости (7.15) для усиления за счет фильтрации

Зависимость $g(\varepsilon)$ от ε показана на рис. 7.3. На рисунке видно, что это ограничение существенно меньше исходного уровня ошибок ε. Рекурсивно применяя алгоритм усиления, уровень ошибок можно сделать сколь угодно малым. Другими словами, модель слабого обучения, производительность которой мало отличается от случайного выбора, при помощи усиления можно преобразовать в сильную модель обучения. Именно в этом смысле можно с полным правом утверждать, что слабая и сильная модели обучения на самом деле эквивалентны.

Алгоритм адаптивного усиления AdaBoost

Практическое ограничение усиления за счет фильтрации состоит в том, что для него часто требуется довольно большое множество примеров. Это ограничение можно обойти, используя другой алгоритм усиления, который называется *AdaBoost* [319], [320] и принадлежит к категории усиления с использованием подвыборки. Контур подвыборки AdaBoost является обычным контуром пакетного обучения. И, что более важно, — он допускает повторное использование данных обучения.

Как и в случае использования алгоритма усиления за счет фильтрации, AdaBoost позволяет работать со слабыми моделями обучения. Целью этого алгоритма является поиск окончательной функции отображения или гипотезы, которая будет иметь низкий уровень ошибок на данном распределении **D** маркированных примеров обучения. От других алгоритмов усиления AdaBoost отличается следующим.

- AdaBoost *адаптивно* настраивается на ошибки слабых гипотез, возвращаемых слабыми моделями обучения. Благодаря именно этому свойству алгоритм и получил свое название.

- Ограничение производительности алгоритма AdaBoost зависит от производительности слабой модели обучения только на тех распределениях, которые фактически формируются в процессе обучения.

Алгоритм AdaBoost работает следующим образом. На итерации n алгоритм *усиления* реализует слабую модель обучения с распределением $\mathbf{D}_n$ множества примеров обучения $\mathbf{T}$. Слабая модель обучения вычисляет гипотезу $\mathbf{F}_n$: $\mathbf{X} \rightarrow Y$, которая корректно классифицирует часть примеров обучения. Ошибка измеряется относительно распределения $\mathbf{D}_n$. Этот процесс продолжается T итераций, в результате чего машина *усиления* объединяет гипотезы $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_T$ в единую заключительную гипотезу $\mathbf{F}_{\text{fin}}$.

Для того чтобы вычислить распределение $\mathbf{D}_n$ на итерации n и заключительную гипотезу $\mathbf{F}_{\text{fin}}$, используется простая процедура, описанная вкратце в табл. 7.2. Исходное распределение является равномерным на множестве примеров обучения $\mathbf{T}$, т.е.

$$\mathbf{D}_1(i) = \frac{1}{n} \text{ для всех } i.$$

Для данного распределения $\mathbf{D}_n$ и слабой гипотезы $\mathbf{F}_n$ на итерации n следующее распределение $\mathbf{D}_{n+1}$ вычисляется умножением веса примера i на некоторое число $\beta_n \in [0, 1)$, если гипотеза $\mathbf{F}_n$ корректно классифицирует входной вектор $\mathbf{x}_i$. В противном случае вес остается неизменным. В итоге все веса заново нормализуются делением на константу нормализации Z_n. В результате "легкие" примеры множества $\mathbf{T}$, корректно классифицированные предыдущей слабой гипотезой, будут иметь более низкий вес, в то время как "трудные" примеры, для которых частота ошибок классификации была высокой, будут иметь больший вес. Таким образом, алгоритм AdaBoost концентрирует основной объем весов на тех примерах, классифицировать которые оказалось сложнее всего.

Окончательная гипотеза $\mathbf{F}_{\text{fin}}$ вычисляется голосованием (т.е. взвешенным линейным порогом) из множества гипотез $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_T$. Это значит, что для данного входного вектора $\mathbf{x}$ окончательная гипотеза дает на выходе метку d, которая максимизирует сумму весов слабых гипотез, которые определили эту метку. Вес гипотезы $\mathbf{F}_n$ определяется как $\log(1/\beta_n)$, т.е. больший вес назначается гипотезе с меньшей ошибкой.

Важное теоретическое свойство алгоритма AdaBoost было сформулировано в следующей теореме [319].

Пусть при вызове алгоритма AdaBoost слабая модель обучения сгенерировала гипотезы с ошибками $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_T$, где ошибка ε_n на итерации n алгоритма AdaBoost определяется как

$$\varepsilon_n = \sum_{i:\mathbf{F}_n(\mathbf{x}_i) \neq d_i} \mathbf{D}_n(i).$$

ТАБЛИЦА 7.2. Краткое описание алгоритма AdaBoost

Дано: Множество примеров обучения $\{(x_i, d_i)\}_{i=1}^{N}$
Распределение $\mathbf{D}$ для N маркированных примеров
Слабая модель обучения
Целое число T, задающее количество итераций алгоритма

Инициализация: $\mathbf{D}_1(i) = 1/N$ для всех i

Вычисления: Для $n = 1, 2, \ldots, T$ выполняются следующие действия.

1. Вызывается слабая модель обучения, реализуемая на распределении $\mathbf{D}_n$
2. Формируется гипотеза $\mathbf{F}_n$: $\mathbf{X} \rightarrow Y$
3. Вычисляется ошибка гипотезы $\mathbf{F}_n$
$$\varepsilon_n = \sum_{i:\mathbf{F}_n(\mathbf{x}_i) \neq d_i} \mathbf{D}_n(i)$$
4. Вычисляется $\beta_n = (\varepsilon_n)/(1 - \varepsilon_n)$
5. Изменяется распределение $\mathbf{D}_n$:
$$\mathbf{D}_{n+1}(i) = \frac{\mathbf{D}_n(i)}{Z_n} \times \begin{cases} \beta_n, & \text{если } \mathbf{F}_n(\mathbf{x}_i) = d_i, \\ 1, & \text{в противном случае,} \end{cases}$$
где Z_n — константа нормализации (выбираемая таким образом, чтобы $\mathbf{D}_{n+1}$ было распределением вероятности)

Выход: Окончательная гипотеза вычисляется следующим образом:
$$\mathbf{F}_n(\mathbf{x}) = \arg\max_{d \in \mathbf{D}} \sum_{n:\mathbf{F}_n(\mathbf{x})=d} \log \frac{1}{\beta_n}$$

Пусть также $\varepsilon_n \leq 1/2$ *и* $\gamma_n = 1/2 - \varepsilon_n$. *Тогда ошибка окончательной гипотезы ограничена сверху следующей величиной:*

$$\frac{1}{N} |\{i : \mathbf{F}_{\text{fin}}(\boldsymbol{x}_i) \neq d_i\}| \leq \prod_{n=1}^{T} \sqrt{1 - 4\gamma_n^2} \leq \exp\left(-2 \sum_{i=1}^{T} \gamma_n^2\right). \tag{7.16}$$

Согласно этой теореме, если ошибка слабой гипотезы, построенной слабой моделью обучения, лишь немного меньше величины 1/2, то ошибка обучения окончательной гипотезы $\mathbf{F}_{\text{fin}}$ экспоненциально стремится к нулю. Однако это совсем не значит, что ошибка обобщения на тестовых примерах тоже будет мала. Эксперименты, представленные в [319], показали следующее. Во-первых, теоретическая граница ошибки обучения часто оказывается слабой. Во-вторых, ошибка обобщения зачастую существенно лучше, нежели это следует из теоретических выкладок.

В табл. 7.2 представлено описание алгоритма AdaBoost для задачи двоичной классификации.

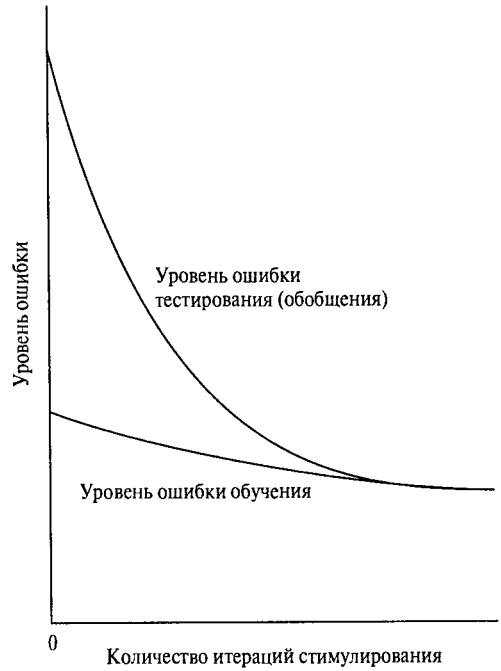


Рис. 7.4. Ошибка алгоритма AdaBoost

Если количество возможных классов (меток) $M>2$, задача усиления становится более интересной ввиду того, что вероятность корректности случайного решения равна $1/M$, что существенно меньше чем 1/2. Для того чтобы алгоритм усиления в такой ситуации мог использовать любую гипотезу, даже слабо отличающуюся от случайного решения, следует несколько модифицировать алгоритм и определение "слабого обучения". Подобные модификации были описаны в [318] и [939].

Изменение ошибки

Эксперименты с алгоритмом AdaBoost, описанные в [153], наглядно показали, что ошибки обучения и тестирования, представленные как функции от количества итераций усиления, имеют следующую зависимость: ошибки тестирования продолжают уменьшаться после того, как ошибки обучения сократились практически до нуля (рис. 7.4). Аналогичный результат был ранее описан в работе, посвященной усилению за счет фильтрации [266].

Эффект, показанный на рис. 7.4, может показаться очень неожиданным в свете того, что нам известно об эффективности обобщения в обычных нейронных сетях. Как известно из главы 4, для многослойного персептрона, обучаемого с помощью алгоритма обратного распространения, ошибка тестирования уменьшается, достигает своего минимума, а затем начинает возрастать в связи с *избыточным обучением*

(overfitting) (см. рис. 4.20). Поведение, показанное на рис. 7.4, в корне отличается тем, что по мере усложнения сети в процессе обучения ошибка обобщения все равно продолжает уменьшаться. Такое поведение противоречит принципу *бритвы Оккама* (Occam's razor), утверждающему, что для обеспечения наилучшего обобщения обучаемая машина должна быть настолько простой, насколько это возможно.

В [941] дано объяснение этому феномену применительно к алгоритму AdaBoost. Ключевой идеей анализа, представленного в работе, является то, что при оценке ошибки обобщения, производимой машиной усиления, следует учитывать не только ошибку обучения, но и *достоверность* (confidence) классификации. Проведенный анализ обнаружил зависимость между усилением и машинами опорных векторов (см. предыдущую главу). В частности, *граница* (margin) классификации определяется как разность между весом, назначенным правильной метке, и максимальным весом, назначенным некоторой некорректной метке. Из этого определения легко увидеть, что граница является числом из диапазона $[-1, +1]$ и образ корректно классифицируется тогда и только тогда, когда его граница имеет положительное значение. Таким образом, в [941] было показано, что явление, наблюдаемое на рис. 7.4, на самом деле связано с распределением границ обучающих примеров по отношению к ошибкам классификации, полученным в результате голосования. Здесь снова можно подчеркнуть, что анализ, проведенный в вышеупомянутой работе, относится только к алгоритму AdaBoost и не применим ни к какому другому алгоритму усиления.

7.5. Компьютерный эксперимент 2

В этом эксперименте мы исследуем алгоритм усиления за счет фильтрации, используемый для решения крайне сложной задачи классификации. Данная задача классификации является двумерной и имеет невыпуклые области решений (рис. 7.5). Первый класс точек лежит в области, помеченной на рисунке $\mathbf{C}_1$, в то время как второй класс — в области $\mathbf{C}_2$. Требуется построить ассоциативную машину, определяющую принадлежность примеров одному из этих двух классов.

Ассоциативная машина, используемая для решения поставленной задачи, состоит из трех экспертов. Каждый из экспертов является многослойным персептроном типа 2-5-2 (два входных узла, пять скрытых и два выходных нейрона). Для обучения используется алгоритм обратного распространения. На рис. 7.6 показан график распределения данных, используемых для обучения всех трех экспертов. На рис. 7.6, *а* показаны данные, используемые для обучения первого эксперта. Данные на рис. 7.6, *б* были отфильтрованы первым экспертом после завершения обучения. Они будут использоваться для обучения второго эксперта. Данные на рис. 7.6, *в* были отфильтрованы в процессе совместной работы первого и второго экспертов. Они будут использоваться для обучения третьего эксперта. Размер множества обучения каждого из экспертов состоял из $N_1 = 1000$ примеров. Из этих трех рисунков видно следующее.

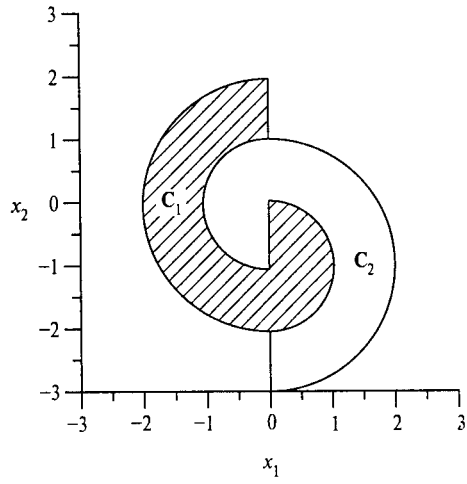


Рис. 7.5. Конфигурация множеств, использованных для моделирования усиления

- Данные, использованные для обучения первого эксперта, распределены равномерно (см. рис. 7.6, *а*).
- Данные, использованные для обучения второго эксперта, более плотно сконцентрированы в областях, помеченных буквами А и Б, т.е. в тех областях, где первый эксперт испытывал сложности при классификации. Количество точек данных в этих двух областях равно количеству правильно классифицированных точек.
- Данные, используемые для обучения третьего эксперта, лежат в области, где первый и второй эксперты испытывали особые сложности при классификации.

На рис. 7.7, *а–в* показаны границы решений, сформированные экспертами 1, 2 и 3 соответственно. На рис. 7.7, *г* показана общая граница решений, сформированная совместными усилиями всех трех экспертов (в данном случае производилось обычное суммирование выходных сигналов отдельных экспертов). Обратите внимание, что различие областей решений первого и второго экспертов (см. рис. 7.7, *а* и *б*) формирует множество точек данных, используемых для обучения третьего эксперта (см. рис. 7.6, *в*).

Вероятность корректной классификации трех экспертов на тестовых данных составила:

1 эксперт: 75,15%
2 эксперт: 71,44%
3 эксперт: 68,90%

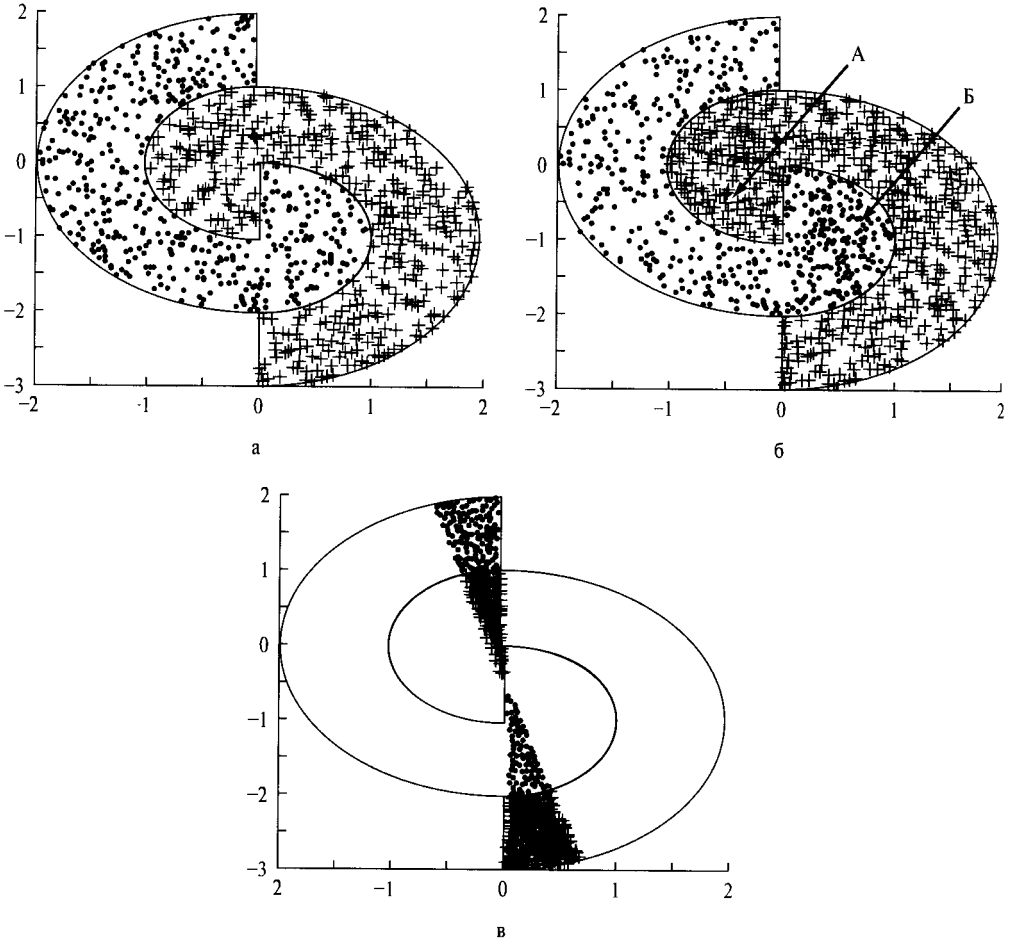


Рис. 7.6. Графики распределения множеств обучения экспертов в компьютерном эксперименте по усилению: эксперт 1 (а); эксперт 2 (б); эксперт 3 (в)

Общая вероятность корректной классификации всей ассоциативной машины составила 91,72%. При вычислении этого показателя использовалось множество тестовых данных, состоящее из 32000 точек. Общая граница решений, сформированная алгоритмом *усиления* с тремя экспертами, показана на рис. 7.7, *г*. Этот рисунок является еще одним доказательством хорошей эффективности классификации.

7.6. Ассоциативная гауссова модель смешения

Вторая часть настоящей главы, которая начинается с этого раздела, будет посвящена изучению второго класса ассоциативных машин — динамических структур. Исполь-

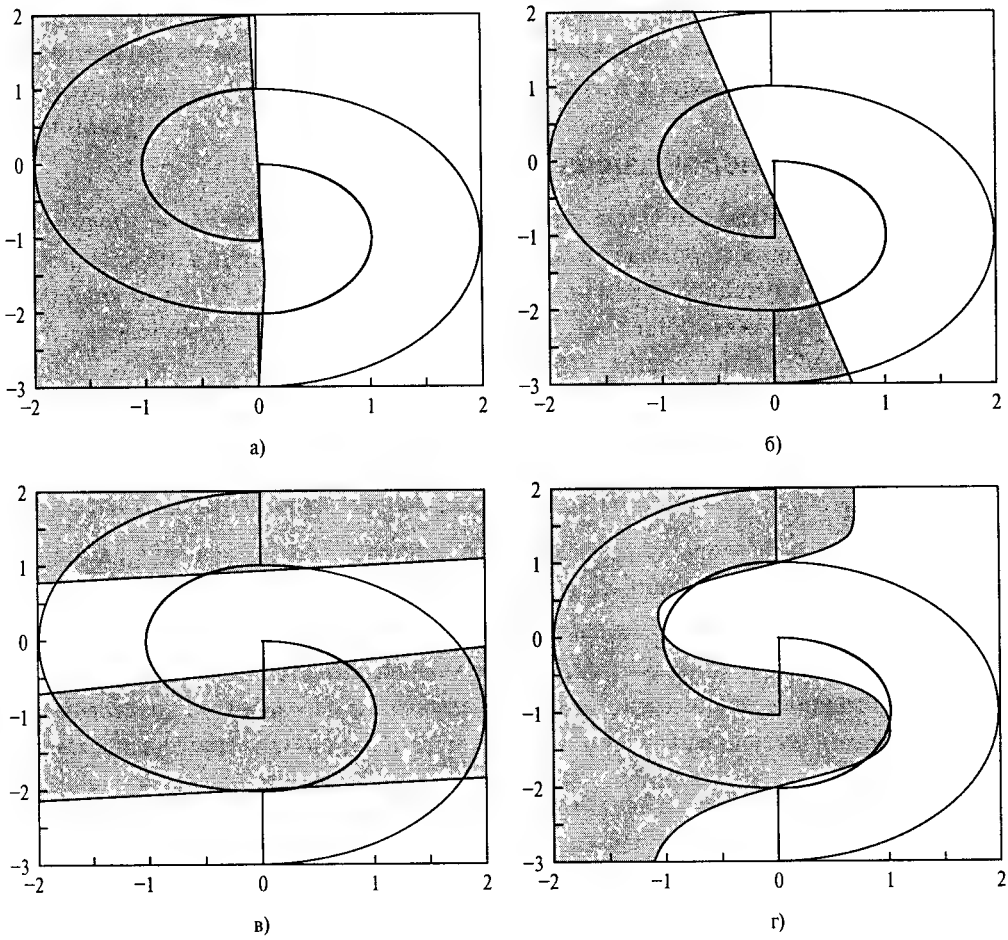


Рис. 7.7. Границы решений, сформированные различными экспертами в эксперименте по усилению: эксперт 1 (а); эксперт 2 (б); эксперт 3 (в); ассоциативная машина в целом (г)

зуемый здесь термин "динамические" подразумевает то, что объединение знаний, накопленных экспертами, происходит при участии самого входного сигнала.

Для того чтобы начать изучение данного вопроса, рассмотрим модульную нейронную сеть, в которой процесс обучения происходит при неявном объединении самоорганизующейся формы обучения и формы обучения с учителем. Эксперты технически обеспечивают обучение с учителем, поскольку их отдельные выходы объединяются для получения желаемого отклика. Однако сами эксперты осуществляют самоорганизующееся обучение. Это значит, что они самоорганизуются с целью нахождения оптимального разбиения входного пространства, причем каждый из них в своем подпространстве имеет наилучшую производительность, а вся группа обеспечивает хорошую модель всего входного пространства.

Описанная схема обучения отличается от схем, рассмотренных в предыдущих трех главах, поскольку здесь для генерации данных обучения предполагается использование специфической модели.

Вероятностная порождающая модель

Для того чтобы понять основную идею, рассмотрим задачу *регрессии* (regression), в которой регрессор $\mathbf{x}$ порождает отклик, обозначаемый случайной переменной D. Реализацию этой случайной переменной обозначим символом d. С целью упрощения выкладок, не ограничивая общности, будем рассматривать скалярную модель регрессии. В частности, предполагается, что генерация отклика d определяется следующей вероятностной моделью [525].

1. Из некоторого наперед заданного распределения случайным образом выбирается вектор $\mathbf{x}$.
2. Для заданного вектора $\mathbf{x}$ и некоторого вектора параметров $\mathbf{a}^{(0)}$ выбирается конкретное (например, k-е) правило в соответствии с условной вероятностью $P(k|\mathbf{x},\mathbf{a}^{(0)})$.
3. Для правила $k, k = 1, 2, \ldots, K$, отклик модели является линейным по $\mathbf{x}$ с аддитивной ошибкой ε_k, моделируемой как случайная переменная с гауссовым распределением, имеющим среднее значение нуль и единичную дисперсию, т.е.

$$E[\varepsilon_k] = 0 \text{ для всех } k \tag{7.17}$$

и

$$\text{var}[\varepsilon_k] = 1 \text{ для всех } k. \tag{7.18}$$

Предположение о единичной дисперсии в п. 3 было сделано из соображений дополнительного упрощения изложения. В общем случае каждый эксперт может иметь отличную от 1 дисперсию выходного сигнала, формируемую на основе данных обучения.

Вероятностная генерация переменной D определяется условной вероятностью $P(D = d|\mathbf{x}, \mathbf{w}_k^{(0)})$ для заданного вектора $\mathbf{x}$ и некоторого вектора параметров $\mathbf{w}_k^{(0)}$, где $k = 1, 2, \ldots, K$. Описанная вероятностная порождающая модель не обязательно должна иметь прямую взаимосвязь с некоторым физическим явлением. Требуется лишь, чтобы реализованные в ней вероятностные решения представляли *абстрактную* модель, которая с возрастающей точностью определяет положение *условного среднего значения отклика d в нелинейном многообразии*, которое связывает входной вектор со средним значением выходного сигнала [521].

Согласно представленной модели, отклик D может быть сгенерирован K различными способами в соответствии с K вариантами выбора метки k. Таким образом, условная вероятность генерирования отклика $D = d$ для данного входного вектора $\mathbf{x}$ будет равна

$$P(D = d|\mathbf{x}, \boldsymbol{\theta}^{(0)}) = \sum_{k=1}^{K} P(D = d|\mathbf{x}, \mathbf{w}_k^{(0)}) P(k|\mathbf{x}, \mathbf{a}^{(0)}), \tag{7.19}$$

где $\boldsymbol{\theta}^{(0)}$ — *вектор параметров порождающей модели* (generative model parameter vector), обозначающий комбинацию $\mathbf{a}^{(0)}$ и $\{\mathbf{w}_k^{(0)}\}_{k=1}^{K}$. Верхний индекс 0 в обозначениях $\mathbf{a}^{(0)}$ и $\mathbf{w}_k^{(0)}$ введен для того, чтобы отличать параметры порождающей модели от параметров модели смешения мнений экспертов, которая рассматривается ниже.

Модель смешения мнений экспертов

Рассмотрим конфигурацию сети, показанную на рис. 7.8. Такая сеть носит название *смешения мнений экспертов* (mixture of experts или ME)[4] и состоит из K модулей, обучаемых с учителем и называемых *сетями экспертов* (expert network), или просто *экспертами*. Интегрирующий элемент носит название *сеть шлюза* (gating network). Он выполняет функцию посредника между сетями экспертов. Предполагается, что различные эксперты лучше всего работают в своих областях входного пространства согласно описанной вероятностной порождающей модели. Исходя из этого и возникает потребность в сети шлюза.

В предположении скалярности задачи регрессии каждая из сетей экспертов представляет собой линейный фильтр. На рис. 7.9 показан граф передачи сигнала одного нейрона, соответствующего эксперту k. Таким образом, выходной сигнал, производимый экспертом k, является скалярным произведением входного вектора $\mathbf{x}$ и вектора синаптических весов $\mathbf{w}_k$ данного нейрона, т.е.

$$y_k = \mathbf{w}_k^T \mathbf{x}, \quad k = 1, 2, \ldots, K. \tag{7.20}$$

Сеть шлюза содержит один слой из K нейронов. Каждый из этих нейронов соответствует одному из экспертов. На рис. 7.10, *а* показан архитектурный граф сети шлюза; на рис. 7.10, *б* показан граф передачи сигнала для отдельного нейрона k этой сети. В отличие от экспертов нейроны сети шлюза являются нелинейными. Их функции активации описываются следующим образом:

---

[4] Идея использования сети экспертов для реализации сложных функций отображений впервые была предложена в [508]. Дальнейшее развитие этой модели было обусловлено предложением, описаным в [789] и рассматривающим конкуреитную адаптацию

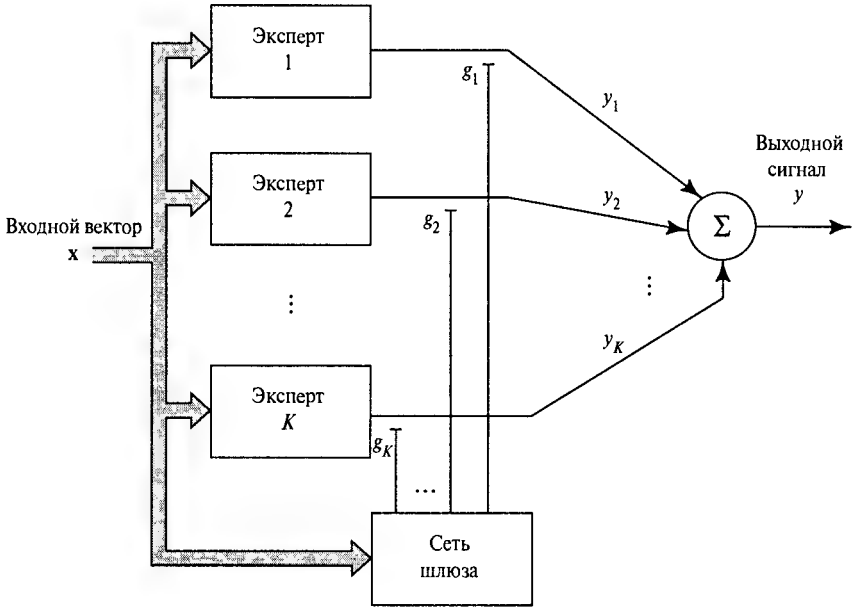


Рис. 7.8. Блочная диаграмма модели ME; скалярные выходы экспертов усреднены сетью шлюза

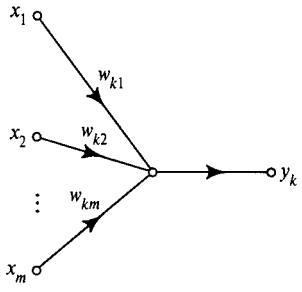


Рис. 7.9. Граф передачи сигнала единичного линейного нейрона, составляющего сеть эксперта k

$$g_k = \frac{\exp(u_k)}{\sum_{j=1}^{K} \exp(u_j)}, \quad k = 1, 2, \ldots, K, \tag{7.21}$$

где u_k — результат скалярного произведения входного вектора $\mathbf{x}$ и вектора синаптических весов $\mathbf{a}_k$, т.е.

$$u_k = \mathbf{a}_k^T \mathbf{x}, \quad k = 1, 2, \ldots, K. \tag{7.22}$$

"Нормализованное" экспоненциальное преобразование (7.21) можно рассматривать как обобщение логистической функции для нескольких входов. В ней сохраня-

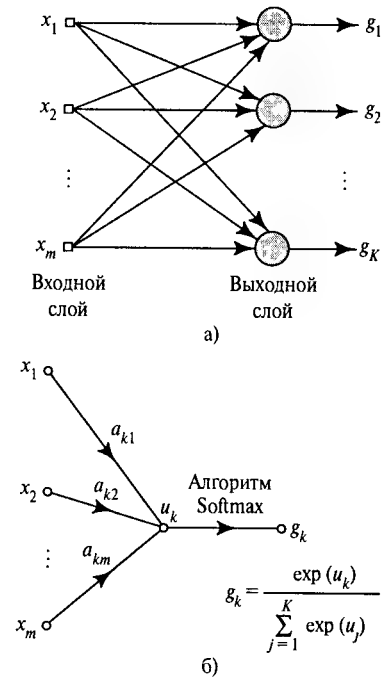


Рис. 7.10. Один слой нейронов сети шлюза (а); граф передачи сигнала для этого нейрона (б)

ется порядок входных значений, но при этом реализовано дифференцируемое обобщение операции "победитель получает все", извлекающей максимальное значение. По этой причине функция активации (7.21) называется *softmax* [155]. Обратите внимание, что линейная зависимость u_k от входного вектора $\mathbf{x}$ приводит к нелинейной зависимости выходного значения сети шлюза от $\mathbf{x}$.

Для вероятностной интерпретации роли сети шлюза рассмотрим ее как "классификатор", который отображает входной вектор $\mathbf{x}$ в значение *мультиномиальной вероятности* (multinomial probability) так, чтобы различные эксперты могли соответствовать желаемому отклику [525].

Важно отметить, что использование функции активации softmax в сети шлюза гарантирует, что эти вероятности будут удовлетворять следующим условиям:

$$0 \leq g_k \leq 1 \text{ для всех } k \tag{7.23}$$

и

$$\sum_{k=1}^{K} g_k = 1. \tag{7.24}$$

Пусть y_k — выходной сигнал k-го эксперта, производимый в ответ на входной вектор $\mathbf{x}$. Тогда общий выход модели ME будет следующим:

$$y = \sum_{k=1}^{K} g_k y_k, \tag{7.25}$$

где, как уже отмечалось ранее, g_k — нелинейная функция $\mathbf{x}$. Допуская, что выбрано правило k вероятностной модели и что входным вектором является $\mathbf{x}$, выход эксперта y_k можно трактовать как условное среднее значение случайной переменной D:

$$E[D|\mathbf{x}, k] = y_k = \mathbf{w}_k^T \mathbf{x}, \quad k = 1, 2, \ldots, K. \tag{7.26}$$

Обозначая символом μ_k условное среднее значение переменной D, можно записать:

$$\mu_k = y_k, k = 1, 2, \ldots, K. \tag{7.27}$$

Дисперсия переменной совпадает с дисперсией ошибки ε_k. Таким образом, используя равенство (7.18), приходим к соотношению:

$$\text{var}[D|\mathbf{x}, k] = 1, k = 1, 2, \ldots, K. \tag{7.28}$$

Функция плотности вероятности переменной D для данного входного вектора $\mathbf{x}$ с учетом предположения о том, что выбрано k-е правило вероятностной порождающей модели (т.е. эксперт k), может быть записана в следующем виде:

$$f_D(d|\mathbf{x}, k, \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(d - y_k)^2\right), \quad k = 1, 2, \ldots, K, \tag{7.29}$$

где $\boldsymbol{\theta}$ — вектор, объединяющий параметры сети шлюза и параметры экспертов модели ME. Функция плотности вероятности переменной D при данном $\mathbf{x}$ является *смесью* функций плотности вероятности $\{f_D(d|\mathbf{x}, k, \boldsymbol{\theta})\}_{k=1}^{K}$, где смешанные параметры являются мультиномиальными вероятностями, определяемыми сетью шлюза. Исходя из этого можно записать следующее:

$$f_D(d|\mathbf{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} g_k f_D(d|\mathbf{x}, k, \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^{K} g_k \exp\left(-\frac{1}{2}(d - y_k)^2\right). \tag{7.30}$$

Распределение вероятности (7.30) называется *ассоциативной гауссовой моделью смешения* (associative Gaussian mixture model) [716], [1057], которая вкратце описывалась в главе 5. Ассоциативная модель отличается от неассоциативной тем, что

условное среднее значение μ_k и параметры смешения g_k *не фиксированы*; все они являются функциями входного вектора $\mathbf{x}$. Таким образом, ассоциативная гауссова модель смешения (7.30) может рассматриваться как обобщение обычной гауссовой модели смешения.

Важными чертами модели ME (см. рис. 7.8) при условии ее адекватного обучения являются следующие.

1. Выход y_k k-го эксперта является оценкой условного среднего значения случайной переменной для данного желаемого отклика D, заданного вектора $\mathbf{x}$ и правила k вероятностной порождающей модели.
2. Выход g_k сети шлюза определяет мультиномиальную вероятность того, что выход эксперта k соответствует значению $D = d$ на основе знаний, полученных только от вектора $\mathbf{x}$.

С учетом распределения вероятности (7.30) и заданного множества примеров обучения $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ задача сводится к *обучению* условного среднего $\mu_k = y_k$ и параметров смешения $g_k, k = 1, 2, \ldots, K$, таким оптимальным образом, чтобы функция $f_D(d|\mathbf{x}, \boldsymbol{\theta})$ представляла собой хорошую оценку функции плотности вероятности среды, отвечающей за генерирование данных обучения.

Пример 7.1

Поверхность регрессии

Рассмотрим модель ME с двумя экспертами и сетью шлюза с двумя выходами, обозначаемыми как g_1 и g_2. Выход g_1 определяется следующей формулой:

$$g_1 = \frac{\exp(u_1)}{\exp(u_1) + \exp(u_2)} = \frac{1}{1 + \exp(-(u_1 - u_2))}. \tag{7.31}$$

Пусть $\mathbf{a}_1$ и $\mathbf{a}_2$ — два вектора весов сети шлюза. Тогда можно записать:

$$u_k = \mathbf{x}^T \mathbf{a}_k, k = 1, 2$$

и, таким образом, переписать выражение (7.31) в следующем виде:

$$g_1 = \frac{1}{1 + \exp(-\mathbf{x}^T(\mathbf{a}_1 - \mathbf{a}_2))}. \tag{7.32}$$

Второй выход сети шлюза можно выразить так:

$$g_2 = 1 - g_1 = \frac{1}{1 + \exp(-\mathbf{x}^T(\mathbf{a}_2 - \mathbf{a}_1))}.$$

Таким образом, оба выхода, g_1 и g_2, имеют форму логистической функции, однако с одним отличием. Ориентация g_1 определяется направлением вектора разности $(\mathbf{a}_1 - \mathbf{a}_2)$, в то время как ориентация g_2 определяется направлением вектора разности $(\mathbf{a}_2 - \mathbf{a}_1)$, т.е. направления векторов g_1 и g_2 противоположны друг другу. Вдоль *хребта* (ridge), определяемого соотношением $\mathbf{a}_1 = \mathbf{a}_2$, имеем $g_1 = g_2 = 1/2$. Таким образом, оба эксперта вносят одинаковый вклад в выход модели ME. В стороне от хребта один из двух экспертов играет доминирующую роль. ■

7.7. Модель иерархического смешения мнений экспертов

Модель ME, изображенная на рис. 7.8, разбивает входное пространство на несколько подпространств. При этом за распределение информации (собранной на основе данных обучения) по отдельным экспертам отвечает одна сеть шлюза. Модель *иерархического смешения мнений экспертов* (hierarchical mixture of experts — HME), представленная на рис. 7.11, представляет собой естественное расширение модели ME. На рисунке показана модель HME с четырьмя экспертами. Архитектура модели HME подобна *дереву*, в котором сети шлюзов являются ветвями, а отдельные эксперты — листьями. Модель HME отличается от модели ME тем, что входное пространство разбивается на множество *вложенных* подпространств, а информация объединяется и перераспределяется между экспертами под управлением нескольких сетей шлюзов, организованных в иерархическую структуру.

Модель HME на рис. 7.11 имеет *два уровня иерархии* (two levels of hierarchy), или *два слоя сетей шлюзов* (two layers of gating networks). Продолжая применять принцип "разделяй и властвуй", способом, аналогичным к проиллюстрированному, можно построить модель HME с несколькими уровнями иерархии. Обратите внимание, что в соответствии с соглашением, принятым на рис. 7.11, нумерация уровней шлюзов начинается от выходного узла дерева.

Модель HME можно формально описать двумя способами [521].

1. *Модель HME является результатом стратегии "разделяй и властвуй"*. Если мы верим, что хорошей стратегией является разбиение входного пространства на области, тогда не менее хорошей стратегией будет деление этих областей на регионы. Эта стратегия может быть продолжена рекурсивно до тех пор, пока мы не достигнем стадии, на которой сложность аппроксимирующих поверхностей не будет соответствовать "локальной" сложности данных обучения. Таким образом, модель HME может работать не хуже, а зачастую и лучше модели ME. И этому факту есть объяснение: более высокий уровень сетей шлюзов модели HME эффективно комбинирует информацию и перераспределяет ее среди экспертов своего поддерева. Следовательно, "сила" каждого из параметров рассматриваемого поддерева используется совместно с другими параметрами, содержащимися в том же поддереве, потенциально улучшая общую производительность модели HME.
2. *Модель HME является деревом мягких решений* (soft-decision). Согласно этой точке зрения, смешение мнений экспертов является всего лишь одноуровневым деревом принятия решений, иногда в шутку называемым *пеньком решений* (decision stump). В более общей постановке модель HME можно рассматривать как вероятностную среду для дерева решений. При этом выходной узел модели HME рассматривается как *корень* дерева. Методология *стандартного* дерева решений

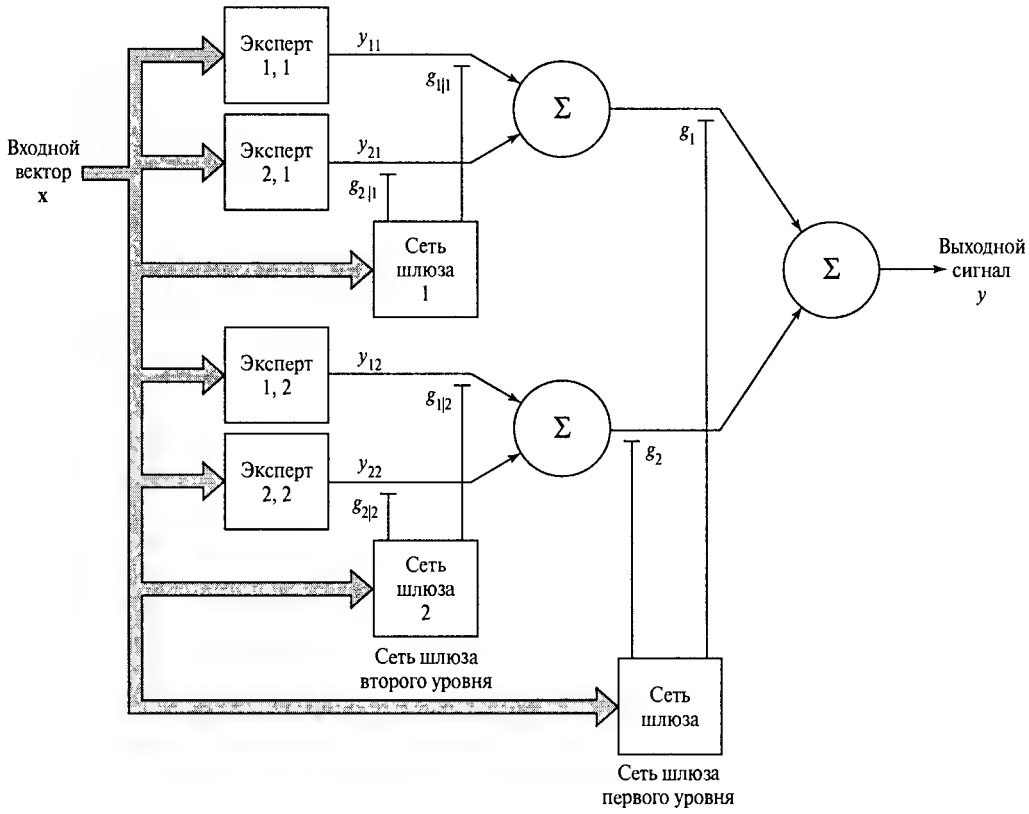


Рис. 7.11. Иерархическое смешение мнений экспертов (HME) для двух уровней иерархии

состоит в построении дерева, ведущего к принятию жестких решений (типа да-нет) в различных областях входного пространства. Противоположностью являются мягкие решения, принимаемые моделью HME. Следовательно, модель HME может превзойти по производительности стандартное дерево решений. Это определяется двумя причинами.

- Жесткие решения, к сожалению, приводят к потере информации, в то время как мягкие решения ее сохраняют. Например, мягкие двоичные решения несут в себе информацию о расстоянии от границы решений (т.е. от точки, в которой решение равновероятно), в то время как жесткие решения — нет. Таким образом, можно утверждать, что, в отличие от стандартного дерева решений, модель обладает встроенным *правилом сохранения информации* (information preservation rule). Это эмпирическое правило утверждает, что информация, содержащаяся во входном сигнале, может быть эффективно сохранена с вычислительной точки зрения до тех пор, пока система не будет готова к окончательному принятию решения или оценке параметра [434].

- Стандартные деревья решений подвержены болезни *жадности* (greediness). Как только в таком дереве принимается решение, оно замораживается и никогда впоследствии не изменяется. Модель HME избавлена от таких проблем, так как решения, принимаемые в дереве, постоянно изменяются. В отличие от стандартного дерева решений в модели HME *можно* отойти от неправильно принятого решения в каком-либо другом месте дерева.

Вторая точка зрения является более предпочтительной. Если рассматривать HME как вероятностный фундамент для построения дерева решений, то можно вычислить подобие для любого предложенного множества данных, а также максимизировать это подобие по параметрам, определяющим разбиение входного пространства на отдельные области. Таким образом, основываясь на информации о стандартных деревьях решений, можно найти практическое решение задачи выбора модели, чем мы и займемся в следующем разделе.

7.8. Выбор модели с использованием стандартного дерева решений

Как и в любой другой нейронной сети, удовлетворительное решение задачи оценки параметров можно получить путем выбора модели, подходящей для конкретной задачи. В случае модели HME под выбором модели понимается выбор количества и композиции узлов решения в дереве. Одно из практических решений задачи выбора модели сводится к запуску на данных обучения алгоритма стандартного дерева решений и принятию полученного дерева в качестве исходного на шаге инициализации в алгоритме обучения, используемом для определения параметров модели HME [521].

Модель HME имеет очевидное сходство со стандартным деревом решений, например с *деревом классификации и регрессии* (classification and regression tree — CART) [154]. На рис. 7.12 показан пример дерева CART, в котором пространство входных данных $\mathbf{X}$ последовательно разбивается серией двоичных делений на *терминальные узлы* (terminal node). При сравнении рис. 7.11 и 7.12 ясно видно, что между моделями CART и HME существует следующее сходство.

- Правило выбора разбиений в промежуточных узлах (ветвях) в модели CART играет роль, аналогичную сетям шлюзов в модели HME.
- Терминальные узлы в модели CART играют роль, аналогичную сетям экспертов в модели HME.

Применяя модель CART для интересующей нас задачи классификации или регрессии, можно использовать преимущества *дискретной* природы этой модели для проведения эффективного поиска среди множества альтернативных деревьев. Используя

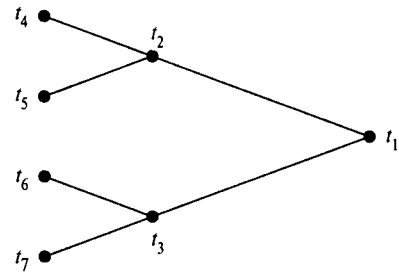


Рис. 7.12. Двоичное дерево решений, описываемое следующим образом: (1) узлы t_2 и t_3 являются потомками узла t_1; (2) узлы t_4 и t_5 являются потомками узла t_2, а узлы t_6 и t_7 являются потомками узла t_3

выбранное таким образом дерево в качестве исходного на шаге инициализации алгоритма обучения, можно взять на вооружение непрерывную вероятностную основу модели HME и получить улучшенную "мягкую" оценку желаемого отклика.

Алгоритм CART

В свете изложенного выше материала приведем краткое описание алгоритма CART. Это описание представлено в контексте задачи регрессии. Имея множество данных обучения $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, алгоритм CART можно использовать для построения двоичного дерева T минимально-квадратичной регрессии. Для этого нужно выполнить следующие действия.

1. *Выбор разбиений* (selection of splits). Пусть узел t — некоторое подмножество дерева T. Пусть $\bar{d}(t)$ — среднее значение d_i для всех пар $(\mathbf{x}_i, d_i)$, попадающих в поддерево t, т.е.

$$\bar{d}(t) = \frac{1}{N(t)} \sum_{\mathbf{x}_i \in t} d_i, \tag{7.33}$$

где суммирование проводится по всем d_i, для которых $\mathbf{x}_i \in t$, а $N(t)$ — общее количество таких случаев. Определим следующие величины:

$$\mathbf{E}(t) = \frac{1}{N} \sum_{\mathbf{x}_i \in t} (d_i - \bar{d}(t))^2 \tag{7.34}$$

и

$$\mathbf{E}(T) = \sum_{t \in T} E(t). \tag{7.35}$$

Для узла t сумма $\sum_{\mathbf{x}_i \in t} (d_i - \bar{d}(t))^2$ представляет собой "внутриузловую сумму квадратов", т.е. общее квадратичное отклонение всех d_i поддерева t от своего среднего

значения $\bar{d}(t)$. Суммирование этих отклонений по всем поддеревьям $t \in T$ дает общую по узлу сумму квадратов, а деление на N — среднее значение.

Для любого заданного множества разбиений S текущего узла t дерева T лучшим разбиением s^* является то, для которого значение $\mathbf{E}(T)$ является наименьшим. Для конкретности предположим, что для любого разбиения s узла t на t_L (новый узел слева от t) и t_R (новый узел справа от t) выполняется соотношение

$$\Delta\mathbf{E}(s,t) = \mathbf{E}(T) - \mathbf{E}(t_L) - \mathbf{E}(t_R). \tag{7.36}$$

Наилучшим разбиением s^* считается то, для которого

$$\Delta\mathbf{E}(s^*,t) = \max_{s\in S} \Delta\mathbf{E}(t,s). \tag{7.37}$$

Таким образом, построенное дерево регрессии будет максимизировать убывание величины $\mathbf{E}(T)$.

2. *Определение терминального узла* (determination of a terminal node). Узел t называется терминальным, если выполняется следующее условие:

$$\max_{s\in S} \Delta\mathbf{E}(t,s) < \beta, \tag{7.38}$$

где β — наперед заданный порог.

3. *Оценка параметров терминального узла по методу наименьших квадратов* (least-squares estimation of a terminal node's parameters). Пусть t является терминальным узлом в двоичном дереве T, а $\mathbf{X}(t)$ — матрица, составленная из $\mathbf{x}_i \in t$. Пусть $\mathbf{d}(t)$ — соответствующий вектор, составленный из желаемых откликов d_i в поддереве t. Определим вектор

$$\mathbf{w}(t) = \mathbf{X}^+(t)\mathbf{d}(t), \tag{7.39}$$

где $\mathbf{X}^+(t)$ — матрица, псевдообратная матрице $\mathbf{X}(t)$. Используя $\mathbf{w}(t)$, на выходе терминального узла t получаем оценку $d(t)$ по методу наименьших квадратов. За счет использования весов, вычисленных согласно (7.39), задача выбора разбиения может быть решена за счет поиска наименьшей суммы квадратов ошибок относительно поверхностей регрессий, а не относительно средних значений.

Использование алгоритма CART для инициализации модели HMA

Предположим, что в результате применения алгоритма CART к множеству данных обучения построено двоичное дерево решений поставленной задачи. Разбиение, построенное алгоритмом CART, можно представить как многомерную поверхность, описываемую формулой

$$\mathbf{a}^T\mathbf{x} + b = 0,$$

где $\mathbf{x}$ — входной вектор; $\mathbf{a}$ — вектор параметров; b — порог.

Рассмотрим в модели HME следующую ситуацию. В примере 7.1 мы обратили внимание, что поверхность регрессии, создаваемая сетью шлюза в двоичном дереве, может быть представлена формулой

$$g = \frac{1}{1 + \exp(-(\mathbf{a}^T\mathbf{x} + b))}, \tag{7.40}$$

описывающей разбиение, в частности при $g = 1/2$. Пусть вектор весов $\mathbf{a}$ для этой конкретной сети шлюза удовлетворяет равенству

$$\mathbf{a} = \|\mathbf{a}\| \cdot \frac{\mathbf{a}}{\|\mathbf{a}\|}, \tag{7.41}$$

где $\|\mathbf{a}\|$ — длина (Евклидова норма) вектора $\mathbf{a}$; $\mathbf{a}/\|\mathbf{a}\|$ — нормализованный вектор (имеющий единичную длину). Подставляя (7.41) в (7.40), параметрическое разбиение в сети шлюза можно переписать следующим образом:

$$g = \frac{1}{1 + \exp\left(-\|\mathbf{a}\|\left(\left(\frac{\mathbf{a}}{\|\mathbf{a}\|}\right)^T\mathbf{x} + \frac{b}{\|\mathbf{a}\|}\right)\right)}. \tag{7.42}$$

Отсюда видно, что вектор $\mathbf{a}/\|\mathbf{a}\|$ задает *направление* разбиения, а $\|\mathbf{a}\|$ — его *резкость* (sharpness). Важно отметить, что сеть шлюза в выражении (7.42), созданная на основе линейного фильтра, за которым следует нелинейная активационная функция softmax, может имитировать разбиение в стиле CART. Более того, мы получаем дополнительную степень свободы — длину вектора параметров $\mathbf{a}$. В стандартном дереве решений дополнительные параметры не нужны, так как для создания разбиения используется жесткое решение. В отличие от алгоритма ME длина вектора $\mathbf{a}$ оказывает существенное влияние на резкость разбиения, выполняемого сетью шлюза в модели HME. В частности, для вектора синаптических весов $\mathbf{a}$ фиксированного направления можно утверждать следующее.

- Если длина вектора **a** достаточно велика (т.е. при низкой температуре), разбиение оказывается резким (sharp).
- Если вектор **a** имеет малую длину (т.е. при высокой температуре), разбиение является более мягким.

Если в пределе $||\mathbf{a}|| = 0$, то разбиение исчезает и по обе стороны исчезнувшего (мысленного) разбиения $g = 1/2$. Эффект от установки длины вектора **a** в значение нуль равнозначен удалению из дерева нетерминального узла, так как рассматриваемая сеть шлюза больше не будет содержать данного разбиения. В предельном случае, когда вектор **a** крайне мал (т.е. при высокой температуре) во всех нетерминальных узлах, вся модель HMA работает подобно одному узлу, т.е. сводится к обычной модели линейной регрессии (в предположении линейности экспертов). По мере увеличения длины векторов синаптических весов сети шлюза модель HME начинает создавать мягкие разбиения, увеличивая доступное для модели количество степеней свободы.

Таким образом, инициализировать модель HME можно следующим образом.

1. Применить алгоритм CART к данным обучения.
2. Установить векторы синаптических весов экспертов модели HME в значения оценок, полученных методом наименьших квадратов для векторов параметров соответствующих терминальных узлов двоичного дерева, построенного в результате применения алгоритма CART.
3. Для сетей шлюзов:
   - задать векторы синаптических весов в соответствии с направлениями, ортогональными соответствующим разбиениям двоичного дерева, полученным в результате алгоритма CART;
   - установить длины (т.е. Евклидовы нормы) векторов синаптических весов равными значениям длин малых случайных векторов.

7.9. Априорные и апостериорные вероятности

Мультиномиальные (multinomial) вероятности g_k и $g_{j|k}$, относящиеся к сетям шлюзов первого и второго уровней, можно рассматривать как *априорные* вероятности в том смысле, что их значения полностью зависимы от входного вектора **x** (возбудителя). Аналогично, можно определить *апостериорные* вероятности h_k и $h_{j|k}$, значения которых зависят как от входного вектора **x**, так и от выходов экспертов в ответ на сигнал **x**. Это последнее множество вероятностей оказывается полезным при разработке алгоритмов обучения для моделей HME.

Возвращаясь к модели HME на рис. 7.11, можно определить апостериорные вероятности в нетерминальных узлах дерева [526]:

$$h_k = \frac{g_k \sum_{j=1}^{2} g_{j|k} \exp\left(-\frac{1}{2}(d - y_{jk})^2\right)}{\sum_{k=1}^{2} g_k \sum_{j=1}^{2} g_{j|k} \exp\left(-\frac{1}{2}(d - y_{jk})^2\right)} \tag{7.43}$$

и

$$h_{j|k} = \frac{g_{j|k} \exp\left(-\frac{1}{2}(d - y_{jk})^2\right)}{\sum_{j=1}^{2} g_{j|k} \exp\left(-\frac{1}{2}(d - y_{jk})^2\right)}. \tag{7.44}$$

Произведение h_k и $h_{j|k}$ определяет *совместную апостериорную вероятность* (joint a posteriori probability) того, что эксперт (j, k) на выходе даст значение y_{jk}, соответствующее желаемому отклику d, т.е.

$$h_{jk} = h_k h_{j|k} = \frac{g_k g_{j|k} \exp\left(-\frac{1}{2}(d - y_{jk})^2\right)}{\sum_{k=1}^{2} g_k \sum_{j=1}^{2} g_{j|k} \exp\left(-\frac{1}{2}(d - y_{jk})^2\right)}. \tag{7.45}$$

Вероятность h_{jk} удовлетворяет следующим двум условиям:

$$0 \leq h_{jk} \leq 1 \text{ для всех } j \text{ и } k \tag{7.46}$$

и

$$\sum_{j=1}^{2} \sum_{k=1}^{2} h_{jk} = 1. \tag{7.47}$$

Применение (7.47) сводится к тому, чтобы распределять доверие к экспертам на конкурентной основе. Более того, из соотношения (7.45) видно, что чем ближе y_{jk} к d, тем более вероятно, что эксперту (j, k) будет дано доверие на соответствие его выхода желаемому отклику, что интуитивно понятно.

Важным свойством модели HME, заслуживающим особого внимания, является *рекурсивность* вычисления *апостериорной* вероятности. Внимательно изучив выражения (7.43) и (7.44), можно заметить, что знаменатель в (7.44) является частью числителя в (7.43). В модели HME нам требуется вычислить апостериорные вероятности для всех нетерминальных узлов дерева решений. Именно поэтому рекурсивность вычислений приобретает исключительно важное практическое значение. В частности, вычисления апостериорных вероятностей всех нетерминальных узлов дерева можно выполнить за один проход.

- При продвижении по дереву в направлении к корневому узлу, уровень за уровнем, апостериорная вероятность любого нетерминального узла вычисляется как сумма *апостериорных* вероятностей всех его "детей".

7.10. Оценка максимального подобия

Переходя к вопросу оценки параметров модели HME, прежде всего хочется заметить, что ее вероятностная интерпретация слегка отличается от модели ME. Так как модель HME формулируется как двоичное дерево, предполагается, что среда, отвечающая за генерацию данных, включает *вложенную последовательность мягких (двоичных) решений, завершающуюся регрессией входного вектора* $\mathbf{x}$ *к выходному сигналу* d. В частности, предполагается, что в *вероятностной порождающей модели* (probabilistic generative model) HME решения моделируются как мультиномиальные случайные переменные [526]. Это значит, что для каждого входного вектора $\mathbf{x}$ величины $g_i(\mathbf{x}, \boldsymbol{\theta}_i^0)$ интерпретируются как мультиномиальные вероятности, связанные с первым решением, а величины $g_{j|i}(\mathbf{x}, \boldsymbol{\theta}_{ji}^0)$ — как условные мультиномиальные распределения, связанные со вторым решением. Как и ранее, верхний индекс 0 обозначает истинные значения параметров порождающей модели. Эти решения формируют дерево решений. Как и в модели ME, в качестве функции активации всех сетей шлюзов модели HME используется softmax. В частности, функция активации g_k k-го выходного нейрона *сети шлюза верхнего уровня* (top-level gating network) имеет вид

$$g_k = \frac{\exp(u_k)}{\exp(u_1) + \exp(u_2)}, \quad k = 1, 2, \tag{7.48}$$

где u_k — взвешенная сумма входных сигналов, поступающих на данный нейрон. Аналогично, функция активации j-го выходного нейрона в k-й сети шлюза второго уровня иерархии описывается следующим образом:

$$g_{j|k} = \frac{\exp(u_{jk})}{\exp(u_{1k}) + \exp(u_{2k})}, \quad (j, k) = 1, 2, \tag{7.49}$$

где u_{jk} — взвешенная сумма входных сигналов, поступающая на данный нейрон. Из соображений простоты выкладок будем рассматривать модель HME всего с двумя уровнями иерархии (т.е. с двумя слоями сетей шлюзов) (рис. 7.11). Как и для модели ME, предполагается, что каждый из экспертов модели HME состоит из одного слоя линейных нейронов. Пусть y_{jk} — выход эксперта (j, k). Тогда общий выход модели HME можно выразить следующим образом:

$$y = \sum_{k=1}^{2} g_k \sum_{j=1}^{2} g_{j|k} y_{jk}. \tag{7.50}$$

Следуя процедуре, аналогичной описанной в разделе 7.6 для модели ME, можно получить функцию плотности вероятности случайной переменной D, представляющей желаемый отклик модели HME, изображенной на рис. 7.11 для заданного вектора $\mathbf{x}$:

$$f_D(d|\mathbf{x},\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^{2} g_k \sum_{j=1}^{2} g_{j|k} \exp\left(-\frac{1}{2}(d - y_{jk})^2\right). \tag{7.51}$$

Таким образом, для заданного множества данных обучения выражение (7.51) определяет модель его распределения. Вектор $\boldsymbol{\theta}$ содержит все синаптические веса, характеризующие сети шлюзов и экспертов модели HME.

Роль *функции подобия* (likelihood function), обозначаемой как $l(\boldsymbol{\theta})$, выполняет функция плотности вероятности $f_D(d|\mathbf{x}, \boldsymbol{\theta})$, рассматриваемая как функция вектора параметров $\boldsymbol{\theta}$. Таким образом, можно записать:

$$l(\boldsymbol{\theta}) = f_D(d|\boldsymbol{x},\boldsymbol{\theta}). \tag{7.52}$$

Хотя функция плотности совместной условной вероятности и функция подобия имеют в точности совпадающие формулы, очень важно отметить физическое различие между ними. В случае функции $f_D(d|\mathbf{x}, \boldsymbol{\theta})$ входной вектор $\mathbf{x}$ и вектор параметров $\boldsymbol{\theta}$ фиксированы, а желаемый отклик d является переменной. В случае функции подобия $l(\boldsymbol{\theta})$ фиксированными являются вектор $\mathbf{x}$ и желаемый отклик d; переменным является только вектор параметров $\boldsymbol{\theta}$.

На практике удобнее работать с натуральным логарифмом функции подобия, а не с самой этой функцией. Для обозначения *логарифмической функции подобия* используется обозначение $L(\boldsymbol{\theta})$ вида

$$L(\boldsymbol{\theta}) = \log[l(\boldsymbol{\theta})] = \log f_D[d|\mathbf{x},\boldsymbol{\theta}]. \tag{7.53}$$

Логарифм функции $l(\boldsymbol{\theta})$ является монотонным преобразованием $l(\boldsymbol{\theta})$. Это значит, что при возрастании функции $l(\boldsymbol{\theta})$ ее логарифм $L(\boldsymbol{\theta})$ также возрастает. Так как $l(\boldsymbol{\theta})$ является функцией плотности условной вероятности, она не может принимать отрицательные значения. Отсюда следует, что при вычислении $L(\boldsymbol{\theta})$ не возникнет никаких проблем. Исходя из этого, оценка $\hat{\boldsymbol{\theta}}$ вектора параметров $\boldsymbol{\theta}$ может быть вычислена как решение *уравнения правдоподобия* (likelihood equation):

$$\frac{\partial}{\partial \boldsymbol{\theta}} l(\boldsymbol{\theta}) = \mathbf{0},$$

или, что эквивалентно, *уравнения логарифмического правдоподобия* (log-likelihood equation):

$$\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}) = \mathbf{0}. \tag{7.54}$$

Термин "максимально правдоподобная оценка" (maximum likelihood estimate) с требуемыми асимптотическими свойствами[5] обычно применяется к корню уравнения правдоподобия, который в глобальном смысле максимизирует функцию подобия $l(\boldsymbol{\theta})$. Однако на практике оценка $\hat{\boldsymbol{\theta}}$ может обеспечивать не глобальный, а только локальный максимум. В любом случае оценка максимального правдоподобия, согласно [296], основывается на сравнительно простой идее.

Разные генеральные совокупности (population) генерируют различные данные, при этом происхождение любого заданного примера более правдоподобно для некоторой определенной совокупности, чем для остальных совокупностей.

Более строго, вектор неизвестных параметров $\boldsymbol{\theta}$ для каждого входного вектора $\mathbf{x}$ оценивается своим *наиболее правдоподобным значением* (most plausible value). Другими словами, оценка максимального правдоподобия $\hat{q}$ является тем значением вектора параметров $\boldsymbol{\theta}$, для которой функция плотности условной вероятности $f_D(d|\mathbf{x}, \boldsymbol{\theta})$ является наибольшей.

---

[5] Алгоритмы оценивания максимального правдоподобия имеют ряд привлекательных особенностей. При достаточно общих условиях можно доказать следующие асимптотические свойства [563].

- Алгоритмы оценивания максимального правдоподобия являются согласованными. Пусть $L(\boldsymbol{\theta})$ – функция логарифмического правдоподобия, а θ_i – некоторый элемент вектора параметров $\boldsymbol{\theta}$. Частная производная $\partial L/\partial \theta_i$ называется счетом (score). Утверждается, что алгоритм оценивания максимального правдоподобия является согласованным, в том смысле, что значение θ_i при счете $\partial L/\partial \theta_i$ равном нулю, сходится по вероятности к истинному значению θ_i при стремлении размера используемого при оценке множества примеров к бесконечности.
- Алгоритмы оценивания максимального правдоподобия являются асимптотически эффективными. Это значит, что
  $\lim\limits_{N\to\infty} \left\{ \frac{\mathrm{var}[\theta_i - \hat{\theta}_i]}{I_{ii}} \right\} = 1$ для всех i,
  где N – размер множества обучения; $\hat{\theta}_i$ – максимально правдоподобная оценка θ_i; I_{ii} – i-й диагональный элемент матрицы, обратной к информационной матрице Фишера
  $$\mathbf{J} = -\begin{bmatrix} E\left[\frac{\partial^2 L}{\partial \theta_1^2}\right] & E\left[\frac{\partial^2 L}{\partial \theta_1 \partial \theta_2}\right] & \dots & E\left[\frac{\partial^2 L}{\partial \theta_1 \partial \theta_M}\right] \\ E\left[\frac{\partial^2 L}{\partial \theta_2 \partial \theta_1}\right] & E\left[\frac{\partial^2 L}{\partial \theta_2^2}\right] & \dots & E\left[\frac{\partial^2 L}{\partial \theta_2 \partial \theta_M}\right] \\ \dots & \dots & \dots & \dots \\ E\left[\frac{\partial^2 L}{\partial \theta_M \partial \theta_1}\right] & E\left[\frac{\partial^2 L}{\partial \theta_M \partial \theta_2}\right] & \dots & E\left[\frac{\partial^2 L}{\partial \theta_M^2}\right] \end{bmatrix}.$$
  где M – размерность вектора параметров $\boldsymbol{\theta}$.
- Алгоритмы оценивания максимального правдоподобия являются асимптотически гауссовыми. Это значит, что при стремлении размера множества примеров к бесконечности каждый элемент оценки максимального правдоподобия $\hat{\theta}$ имеет гауссово распределение.
  На практике оказывается, что асимптотические (т.е. выполняющиеся при больших размерах множества примеров) свойства алгоритмов оценивания максимального подобия достаточно хорошо выполняются при $N \geq 50$.

7.11. Стратегии обучения для модели HME

Вероятностное описание модели HME в разделе 7.10 привело к построению функции логарифмического подобия $L(\boldsymbol{\theta})$ как объекта максимизации. Остается нерешенным вопрос: как осуществить эту максимизацию. Следует отметить, что не существует единого подхода к решению задач максимизации. Существует несколько различных подходов, два из которых описываются в настоящем разделе [507], [526].

1. Подход на основе стохастического градиента (stochastic gradient approach). Этот подход обеспечивает алгоритм для максимизации $L(\boldsymbol{\theta})$ в реальном времени. Его формулировка для двухуровневой модели (см. рис. 7.11) базируется на следующих формулах.

- Вектор градиента $\partial L/\partial \mathbf{w}_{jk}$ для вектора синаптических весов эксперта (j,k).
- Вектор градиента $\partial L/\partial \mathbf{a}_k$ для вектора синаптических весов выходного нейрона k сети шлюза верхнего уровня.
- Вектор градиента $\partial L/\partial \mathbf{a}_{jk}$ для вектора синаптических весов выходного нейрона сети шлюза второго уровня, связанной с экспертом (j,k).

Достаточно просто показать, что (см. задачу 7.9):

$$\frac{\partial L}{\partial \mathbf{w}_{jk}} = h_{j|k}(n)h_k(n)(d(n) - y_{jk}(n))\mathbf{x}(n), \tag{7.55}$$

$$\frac{\partial L}{\partial \mathbf{a}_k} = (h_k(n) - g_k(n))\mathbf{x}(n), \tag{7.56}$$

$$\frac{\partial L}{\partial \mathbf{a}_{jk}} = h_k(n)(h_{j|k}(n) - g_{j|k}(n))\mathbf{x}(n). \tag{7.57}$$

Равенство (7.55) означает, что в процессе обучения синаптические веса эксперта (j,k) изменяются с целью коррекции ошибки между выходом y_{jk} и желаемым откликом d пропорционально совместной *апостериорной* вероятности h_{jk} того, что выход эксперта (j,k) соответствует желаемому отклику d. Равенство (7.56) утверждает, что синаптические веса выходного нейрона k сети шлюза верхнего уровня настраиваются так, чтобы *априорные* вероятности $g_k(n)$ смещались в сторону соответствующих *апостериорных* вероятностей $h_k(n)$. Равенство (7.57) означает, что синаптические веса выходного нейрона сети шлюза второго уровня, связанной с экспертом (j,k), настраиваются с целью коррекции ошибки между *априорной* вероятностью $g_{j|k}$ и соответствующей *апостериорной* вероятностью $h_{j|k}$ пропорционально *апостериорной* вероятности $h_k(n)$.

В соответствии с равенствами (7.55)–(7.57) синаптические веса модели HME изменяются после представления ей каждого примера (возбуждения). Суммируя векторы градиентов по n, можно сформулировать пакетную версию метода градиентного спуска для максимизации функции логарифмического подобия $L(\boldsymbol{\theta})$.

2. Подход на основе максимизации ожидания (expectation-maximization approach — EM). Алгоритм максимизации ожидания (EM), согласно [252], реализует итеративную процедуру вычисления оценки максимального правдоподобия в ситуациях, когда эта проблема решается очень легко. Алгоритм EM получил свое название из-за того, что каждая итерация алгоритма состоит из двух шагов.

- *Шаг ожидания* (expectation step), на котором множество наблюдаемых данных *неполной задачи* (incomplete data problem) и текущее значение вектора параметров используются для получения расширенного *полного набора данных* (complete data set).
- *Шаг максимизации* (maximizing step) заключается в вычислении новой оценки вектора параметров путем максимизации функции логарифмического подобия полного множества данных, созданного на первом шаге.

Таким образом, начиная с подходящего значения вектора параметров, эти действия повторяются поочередно до полной сходимости.

Алгоритм EM оказывается полезным не только в тех случаях, когда набор данных явно не полон, но также и во многих других ситуациях, в которых неполнота данных является не столь очевидной или естественной. В самом деле, вычисление оценки максимального правдоподобия часто облегчается ее формулировкой как задачи с неполными данными. Это происходит потому, что алгоритм EM позволяет использовать пониженную сложность оценки максимального правдоподобия при неполных данных [717]. Модель HME является одним из таких примеров применения. В данном случае отсутствующие данные в виде некоторых переменных-индикаторов искусственно вводятся в модель HME для построения оценки максимального правдоподобия неизвестного вектора параметров (см. раздел 7.12).

Важными особенностями модели HME (при любом подходе — максимизации ожидания или стохастического градиента) являются следующие.

- Все сети шлюзов в модели последовательно вычисляют апостериорные вероятности для всех точек данных множества примеров.
- Корректировки синаптических весов экспертов и сетей шлюзов модели при переходе к следующей итерации являются функциями апостериорной вероятности и соответствующей ей априорной вероятности.

Следовательно, если сеть эксперта, расположенная ниже по дереву, не справляется с задачей обобщения данных в своей локальной окрестности, то происходит смещение регрессионной (дискриминантной) поверхности сети шлюза, расположенной выше в этом дереве. В свою очередь, это смещение помогает экспертам на следующей итерации алгоритма обучения лучше аппроксимировать (fit) данные путем смещения соответствующих подпространств данных. За счет этого модель HME улучшает эффективность "жадного" алгоритма решения, свойственного стандартному дереву решений, например CART.

7.12. Алгоритм EM

Алгоритм EM занимает особое место благодаря своей простоте и общности лежащей в его основе теории, а также широкой области применения[6]. В этом разделе представлено описание алгоритма EM в общем виде. В последующих разделах мы продолжим рассмотрение его применения в задаче оценки параметров модели HME.

Пусть вектор $\mathbf{z}$ состоит из отсутствующих или ненаблюдаемых данных, а $\mathbf{r}$ — вектор с полными данными, составленный из наблюдений d и вектора отсутствующих данных $\mathbf{z}$. Таким образом, существуют два пространства — $\mathbf{D}$ и $\mathbf{R}$, а также отображение пространства $\mathbf{R}$ в $\mathbf{D}$ типа "много к одному". Однако вместо наблюдения полного вектора данных $\mathbf{r}$, фактически можно наблюдать только неполные данные $d = d(\mathbf{r})$ в пространстве $\mathbf{D}$.

Пусть $f_c(\mathbf{r}|\boldsymbol{\theta})$ — функция плотности условной вероятности $\mathbf{r}$ для данного вектора параметров $\boldsymbol{\theta}$. Отсюда следует, что функция плотности условной вероятности случайной переменной D для данного вектора $\boldsymbol{\theta}$ определяется следующим образом:

$$f_D(d|\boldsymbol{\theta}) = \int_{\mathbf{R}(d)} f_c(\mathbf{r}|\boldsymbol{\theta})d\mathbf{r}, \tag{7.58}$$

где $\mathbf{R}(d)$ — подпространство $\mathbf{R}$, определяемое равенством $d = d(\mathbf{r})$. Алгоритм EM направлен на поиск значения $\boldsymbol{\theta}$, максимизирующего *функцию логарифмического правдоподобия на неполных данных* (incomplete-data log-likelihood function):

$$L(\boldsymbol{\theta}) = \log f_D(d|\boldsymbol{\theta}).$$

Эта задача косвенно решается итеративным применением *функции логарифмического правдоподобия на полных данных* (complete-data log-likelihood function):

$$L_c(\boldsymbol{\theta}) = \log f_c(\mathbf{r}|\boldsymbol{\theta}), \tag{7.59}$$

которая является случайной переменной ввиду того, что отсутствующий вектор данных $\mathbf{z}$ не известен.

---

[6] Работу, рассматривающую оценку параметров смеси двух инвариантных гауссовых распределений [779], можно считать самой ранней литературной ссылкой по теме процессов типа EM.

Название "EM-алгоритм" было введено в [252]. В этой фундаментальной работе впервые была представлена формулировка алгоритма EM для вычисления оценок максимального правдоподобия на неполных множествах данных с различными уровнями общности.

Первый объединенный доклад по теории, методологии и применениям алгоритма EM, по его истории и расширениям был представлен в сборнике, выпущенном в 1997 году [717].

Более строго, пусть $\hat{\boldsymbol{\theta}}(n)$ — значение вектора параметров $\boldsymbol{\theta}$ на итерации n алгоритма EM. На E-шаге этой итерации вычисляем математическое ожидание

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) = E[L_c(\boldsymbol{\theta})] \tag{7.60}$$

по $\hat{\boldsymbol{\theta}}(n)$. На M-шаге этой же итерации вычисляем максимум функции $Q(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}(n))$ по $\boldsymbol{\theta}$ в пространстве параметров (весов) $\mathbf{W}$ и, таким образом, находим измененную оценку вектора параметров $\hat{\boldsymbol{\theta}}(n+1)$:

$$\hat{\boldsymbol{\theta}}(n+1) = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)). \tag{7.61}$$

Этот алгоритм начинается с некоторого начального значения $\hat{\boldsymbol{\theta}}(0)$ вектора параметров $\boldsymbol{\theta}$. Затем оба шага последовательно повторяются в соответствии с выражениями (7.60) и (7.61) соответственно, пока разность между $L(\hat{\boldsymbol{\theta}}(n+1))$ и $L(\hat{\boldsymbol{\theta}}(n))$ не окажется меньше некоторого малого наперед заданного значения. В этой точке работа алгоритма завершается.

Заметим, что после каждой итерации алгоритма EM функция логарифмического правдоподобия на неполных данных *не* убывает, т.е. (см. задачу 7.10):

$$L(\hat{\boldsymbol{\theta}}(n+1)) \quad \geq \quad L(\hat{\boldsymbol{\theta}}(n)) \text{ для } n = 0, 1, \ldots. \tag{7.62}$$

Выполнение равенства в этой формуле означает, что мы находимся в стационарной точке функции логарифмического правдоподобия[7].

7.13. Применение алгоритма EM к модели HME

Ознакомившись с алгоритмом EM, можно приступать к решению задачи оценивания параметров модели HME[8].

---

[7] При достаточно общих условиях подобные значения, вычисленные по алгоритму EM, сходятся к стационарным значениям. В [1167] проводится детальное исследование свойств сходимости алгоритма EM. Однако этот алгоритм *не всегда* приводит к локальному или глобальному максимуму функции правдоподобия. В главе 3 [717] представлены два примера, подтверждающие этот факт. В первом из них алгоритм сходился к седловой точке, а во втором — к локальному *минимуму* функции правдоподобия.

[8] С помощью алгоритма EM можно также вычислить байесовскую апостериорную оценкумаксимума (Bayesian maximum a posterior estimation) за счет использования априорной информации для вектора параметров (см. задачу 7.11). Используя правило Байеса, функцию плотности условиой вероятности вектора параметров $\boldsymbol{\theta}$ для заданного множества наблюдеиий $\mathbf{x}$ можно выразить так:

$f_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\mathbf{x}) = \frac{f_{\mathbf{x}}(\mathbf{x}|\boldsymbol{\theta}) f_{\boldsymbol{\theta}}(\boldsymbol{\theta})}{f_{\mathbf{x}}(\mathbf{x})}$.

Из этого соотношения ясно видно, что максимизация апостериориой плотности $f_{\boldsymbol{\theta}}(\boldsymbol{\theta},\mathbf{x})$ эквивалентна максимизации произведеиия $f_{\mathbf{x}}(\mathbf{x}|\boldsymbol{\theta}) f_{\boldsymbol{\theta}}(\boldsymbol{\theta})$, так как $f_{\mathbf{x}}(\mathbf{x})$ не зависит от $\boldsymbol{\theta}$. Функцня плотности вероятности $f_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ представляет собой доступную априорную информацию о $\boldsymbol{\theta}$. Максимизация $f_{\boldsymbol{\theta}}(\boldsymbol{\theta},\mathbf{x})$ дает наиболее вероятную оценку вектора параметров $\boldsymbol{\theta}$ для данного вектора $\mathbf{x}$. В контексте этой оценки важно подчеркнуть следующие моменты.

Пусть $g_k^{(i)}$ и $g_{j|k}^{(i)}$ — (условные) мультиномиальные вероятности, соответствующие решениям, принимаемым сетями шлюзов первого уровня k и второго уровня (j,k) соответственно (см. рис. 7.11), при обработке примера i из обучающего множества. Тогда из выражения (7.51) видно, что соответствующие значения функций плотности условной вероятности случайной переменной D для данного примера $\mathbf{x}_i$ и вектора параметров $\boldsymbol{\theta}$ задаются следующей формулой:

$$f_D(d_i|\mathbf{x}_i,\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^{2} g_k^{(i)} \sum_{j=1}^{2} g_{j|k}^{(i)} \exp\left(-\frac{1}{2}(d_i - y_{jk}^{(i)})^2\right), \tag{7.63}$$

где $y_{jk}^{(i)}$ — выходной сигнал эксперта (j,k) в ответ на подачу i-го примера из множества обучения. Предположим, что все N примеров, содержащихся в обучающем множестве, являются статистически независимыми. Тогда можно определить функцию логарифмического подобия для задачи с неполными данными следующего вида:

$$L(\boldsymbol{\theta}) = \log\left[\prod_{i=1}^{N} f_D(d_i|\mathbf{x}_i,\boldsymbol{\theta})\right]. \tag{7.64}$$

Подставляя выражение (7.63) в (7.64) и игнорируя константу $-(1/2)\log(2\pi)$, получим:

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log\left[\sum_{k=1}^{2} g_k^{(i)} \sum_{j=1}^{2} g_{j|k}^{(i)} \exp\left(-\frac{1}{2}(d_i - y_{jk}^{(i)})^2\right)\right]. \tag{7.65}$$

Чтобы вычислить оценку максимального правдоподобия для вектора $\boldsymbol{\theta}$, необходимо найти стационарную точку (т.е. локальный или глобальный максимум) функции $L(\boldsymbol{\theta})$. К сожалению, функция логарифмического подобия $L(\boldsymbol{\theta})$, согласно формуле (7.65), еще не подходит для такого рода вычислений.

Чтобы обойти эту сложность, *искусственно* дополним множество данных наблюдений $\{d_i\}_{i=1}^N$ множеством отсутствующих данных, полученных в результате выполнения алгоритма EM. Это можно сделать путем введения *переменных-индикаторов* (indicator variable), относящихся к вероятностной модели архитектуры HME, следующим образом [526].

Оценка максимального правдоподобия, представленная максимизацией $f_{\boldsymbol{\theta}}(\boldsymbol{\theta},\mathbf{x})$ по $\boldsymbol{\theta}$, является сокращенной формой максимизации апостериорной вероятности (под словом "сокращенная" понимается отсутствие априорной информации).

Использование априорной информации является синонимом регуляризации, которая (см. главу 5) соответствует гладкому отображению входа на выход.

В [1114] предстаален байесовский подход к оценке параметров модели смеси экспертов, преодолевающий так называемое явление избыточного обучения, которое приводит к оценке с большой дисперсией при использовании принципа максимума подобия.

- $z_k^{(i)}$ и $z_{j|k}^{(i)}$ будем интерпретировать как метки соответствующие решениям, принимаемым вероятностной моделью в ответ на подачу i-го примера множества обучения. Эти переменные определяются таким образом, чтобы при любом значении i только одно из значений $z_k^{(i)}$ и только одно из значений $z_{j|k}^{(i)}$ равнялось единице. Обе переменные — $z_k^{(i)}$ и $z_{j|k}^{(i)}$ — можно рассматривать как статистически независимые дискретные случайные переменные, математическое ожидание которых равно соответственно

$$E[z_k^{(i)}] = P\left[z_k^{(i)} = 1|\mathbf{x}_i, d_i, \hat{\boldsymbol{\theta}}(n)\right] = h_k^{(i)} \tag{7.66}$$

и

$$E[z_{j|k}^{(i)}] = P\left[z_{j|k}^{(i)} = 1|\mathbf{x}_i, d_i, \hat{\boldsymbol{\theta}}(n)\right] = h_{j|k}^{(i)}, \tag{7.67}$$

где $\hat{\boldsymbol{\theta}}(n)$ — оценка параметра $\boldsymbol{\theta}$ на итерации n алгоритма EM.

- $z_{jk}^{(i)} = z_{j|k}^{(i)} z_k^{(i)}$ интерпретируется как метка, определяющая эксперта (j, k) в вероятностной модели для i-го примера множества обучения. Ее также можно трактовать как дискретную случайную величину со следующим математическим ожиданием:

$$E[z_{jk}^{(i)}] = E[z_{j|k}^{(i)} z_k^{(i)}] = E[z_{j|k}^{(i)}]E[z_k^{(i)}] = h_{j|k}^{(i)} h_k^{(i)} = h_{jk}^{(i)}. \tag{7.68}$$

Значения $h_k^{(i)}, h_{j|k}^{(i)}$ и $h_{jk}^{(i)}$ в равенствах (7.66) и (7.68) являются апостериорными вероятностями, введенными в разделе 7.9. Верхний индекс i введен для идентификации рассматриваемого примера в обучающем множестве. В задаче 7.13 читателю будет предложено обосновать все эти три равенства.

С добавлением определенных таким образом отсутствующих данных к множеству наблюдений задача вычисления оценки максимального правдоподобия существенно упрощается. Пусть $f_c(d_i, z_{jk}^{(i)}|\mathbf{x}_i, \boldsymbol{\theta})$ — функция плотности условной вероятности на полном множестве данных, составленная для d_i и $z_{jk}^{(i)}$ при заданных $\mathbf{x}_i$ и векторе параметров $\boldsymbol{\theta}$. Тогда можно записать:

$$f_c(d_i, z_{jk}^{(i)}|\mathbf{x}_i, \boldsymbol{\theta}) = \prod_{j=1}^{2}\prod_{k=1}^{2}(g_k^{(i)} g_{j|k}^{(i)}(i) f_{jk}(d_i)), \tag{7.69}$$

где $f_{jk}(d_i)$ — функция плотности условной вероятности d_i для выбранного эксперта (j, k) в модели HME, т.е. $f_{jk}(i)$ выбирается из гауссова распределения

$$f_{jk}(d_i) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(d_i - y_{jk}^{(i)})^2\right). \tag{7.70}$$

Обратите внимание, что формула (7.69) соответствует гипотетическому эксперименту, содержащему переменные-индикаторы, представленные переменными $z_{jk}^{(i)}$, не существующими в физическом смысле. В любом случае функция логарифмического правдоподобия для задачи с полными данными, включающими в себя все множество примеров обучения, выглядит следующим образом:

$$\begin{aligned} L_c(\boldsymbol{\theta}) &= \log\left[\prod_{i=1}^{N} f_c(d_i, z_{jk}^{(i)} | \mathbf{x}_i, \boldsymbol{\theta})\right] = \\ &= \log\left[\prod_{i=1}^{N}\prod_{j=1}^{2}\prod_{k=1}^{2} \left(g_k^{(i)} g_{j|k}^{(i)}(i) f_{jk}(d_i)\right)^{z_{jk}^{(i)}}\right] = \\ &= \sum_{i=1}^{N}\sum_{j=1}^{2}\sum_{k=1}^{2} z_{jk}^{(i)} \left[\log g_k^{(i)} + \log g_{j|k}^{(i)} + \log f_{jk}(d_i)\right]. \end{aligned} \tag{7.71}$$

Подставляя выражение (7.70) в (7.71) и игнорируя константу $-(1/2)\log(2\pi)$, можно записать:

$$L_c(\boldsymbol{\theta}) = \sum_{i=1}^{N}\sum_{j=1}^{2}\sum_{k=1}^{2} z_{jk}^{(i)} \left[\log g_k^{(i)} + \log g_{j|k}^{(i)} - \frac{1}{2}(d_i - y_{jk}^{(i)})^2\right]. \tag{7.72}$$

Сравнивая выражения (7.72) и (7.65), можно заметить, что введение в множество наблюдений переменных-индикаторов вместо отсутствующих данных дало следующий вычислительный эффект: задача вычисления оценки максимального правдоподобия была разделена на множество задач регрессии для отдельных экспертов и отдельное множество задач классификации для сетей шлюзов.

Для продолжения работы с алгоритмом EM нужно в первую очередь использовать E-шаг этого алгоритма, вычисляя математическое ожидание функции логарифмического подобия на полных данных $L_c(\boldsymbol{\theta})$:

$$\begin{aligned} Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) &= E[L_c(\boldsymbol{\theta})] = \\ &= \sum_{i=1}^{N}\sum_{j=1}^{2}\sum_{k=1}^{2} E\left[z_{jk}^{(i)}\right] \cdot \left[\log g_k^{(i)} + \log g_{j|k}^{(i)} - \frac{1}{2}(d_i - y_{jk}^{(i)})^2\right], \end{aligned} \tag{7.73}$$

где оператор ожидания действует на переменную индикатора $z_{jk}^{(i)}$ как на единственную ненаблюдаемую переменную. Тогда, подставляя (7.68) в (7.73), получим:

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) = \sum_{i=1}^{N} \sum_{j=1}^{2} \sum_{k=1}^{2} h_{jk}^{(i)} \cdot \left[\log g_k^{(i)} + \log g_{j|k}^{(i)} - \frac{1}{2}(d_i - y_{jk}^{(i)})^2 \right]. \tag{7.74}$$

М-шаг этого алгоритма требует максимизации функции $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n))$ по $\boldsymbol{\theta}$. Вектор параметров $\boldsymbol{\theta}$ состоит из двух множеств синаптических весов: одно из них относится к сетям шлюзов, второе — к экспертам. Из предыдущего обсуждения можно сделать следующие выводы.

- Синаптические веса экспертов определяют $y_{jk}^{(i)}$, которые также входят в определение $h_{jk}^{(i)}$. Таким образом, на выражение для $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n))$ эксперты влияют только в части слагаемого $h_{jk}^{(i)}(d_i - y_{jk}^{(i)})^2$.
- Синаптические веса сетей шлюзов определяют вероятности $g_{jk}^{(i)}$, $g_{j|k}^{(i)}$ и $h_{jk}^{(i)}$. Таким образом, на выражение для $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n))$ эксперты влияют только в части слагаемого $h_{jk}^{(i)}(\log g_k^{(i)} + \log g_{j|k}^{(i)})$.

Следовательно, М-шаг алгоритма EM сводится к следующей тройственной задаче оптимизации модели HME с двумя уровнями иерархии:

$$\mathbf{w}_{jk}(n+1) = \arg\min_{\mathbf{w}_{jk}} \sum_{i=1}^{N} h_{jk}^{(i)}(d_i - y_{jk}^{(i)})^2, \tag{7.75}$$

$$\mathbf{a}_j(n+1) = \arg\min_{\mathbf{a}_j} \sum_{i=1}^{N} \sum_{k=1}^{2} h_{jk}^{(i)} \log g_k^{(i)} \tag{7.76}$$

и

$$\mathbf{a}_{jk}(n+1) = \arg\min_{\mathbf{a}_{jk}} \sum_{i=1}^{N} \sum_{l=1}^{2} h_l^{(i)} \sum_{m=1}^{2} h_{m|l}^{(i)} \log g_{m|l}^{(i)}. \tag{7.77}$$

Задачи оптимизации (7.75)–(7.77) решаются при фиксированных h; h является функцией параметров, но производные по h не вычисляются. Обратите внимание, что величины в правых частях выражений относятся к измерениям, выполняемым на шаге n.

Оптимизация выражения (7.75), относящегося к экспертам, представляет собой задачу нахождения взвешенной оценки по методу наименьших квадратов. Остальные две задачи оптимизации — (7.76) и (7.77) — относятся к сетям шлюзов и представляют собой задачи поиска оценки максимального правдоподобия[9].

Несмотря на то что уравнения сформулированы только для двух уровней иерархии, они могут быть расширены для любого их количества.

---

[9] Для решения задачи вычнсления оценки максимального правдоподобия, описываемой уравнениями (7.76) и (7.77), часто используют алгоритм, известный под названием итеративно взвешиваемый алгоритм наименьших квадратов

7.14. Резюме и обсуждение

При изучении задач моделирования, распознавания образов и регрессии необходимо рассматривать два экстремальных случая.

1. *Простые модели*, которые дают представление о данной задаче, но не отличаются особой точностью.
2. *Сложные модели*, которые дают точные результаты, не слишком углубляясь в саму задачу.

Похоже, в рамках одной модели невозможно объединить простоту и точность. В контексте дискуссии, представленной во второй части настоящей главы, алгоритм CART является примером простой модели, которая использует жесткие решения при построении разбиения входного пространства на множество подпространств, каждое из которых имеет собственного эксперта. К сожалению, использование результатов жестких решений приводит к потере части информации и, таким образом, потерям производительности. С другой стороны, многослойный персептрон (MLP) является сложной моделью с вложенной формой нелинейности, создаваемой для сохранения информации из множества обучающих данных. Однако в нем используется подход "черного ящика" для глобального построения единой функции аппроксимации обучающих данных. При этом теряется глубина взгляда на саму задачу. Модель HME, представляющая собой динамический тип ассоциативной машины, обладая преимуществами как CART, так и MLP, является компромиссом между этими двумя экстремальными случаями.

- Архитектура HME аналогична архитектуре CART, но отличается от нее мягкостью разбиения входного пространства.
- HME использует вложенную форму нелинейности, аналогичную MLP, но не для построения отображения входного сигнала в выходной, а с целью разбиения входного пространства.

В этой главе мы заострили внимание на использовании двух методов построения модели HME.

- Алгоритм CART составляет архитектурную основу для задачи выбора модели.
- Алгоритм EM используется для решения задачи оценки параметров. Он итеративно вычисляет оценки максимального правдоподобия для параметров модели.

Алгоритм EM обычно обеспечивает движение вверх по склону функции правдоподобия. Таким образом, инициализируя алгоритм EM с помощью CART так, как описано в разделе 7.8, можно ожидать лучшего возможного качества обобщения первого из них, применяя результаты второго в качестве исходного состояния.

Алгоритм EM является важным и основополагающим в тех областях, где на первое место выходит построение оценки максимального правдоподобия, например в *моделировании*. Интересное приложение моделирования было описано в [509], где модель ME обучалась решению задачи "что–где". В этой задаче модель должна определить, что представляет собой объект и где он находится в визуальном поле. При обучении использовались два эксперта, каждый из которых специализировался на одном из аспектов задачи. Для заданного входного сигнала оба эксперта генерировали выходные. При этом сеть шлюза принимала решение относительно смешения выходов. Успешные результаты, полученные в этой работе, показали, что задачи можно естественным образом распределять, но не на основе принципа "одна задача на всех", а на основе соответствия требований задачи вычислительным свойствам модели [282].

В завершение обсуждения вернемся к еще одному классу ассоциативных машин, о котором говорилось в первой части настоящей главы. В то время как модели HME и ME основаны на использовании сетей шлюзов, активизируемых входными сигналами для объединения знаний, накопленных разными экспертами модели, ассоциативные машины основаны на усреднении по ансамблю или, как альтернатива, на усилении, основанном на интеграции самим алгоритмом обучения.

1. Усреднение по ансамблю улучшает качество обучения в смысле снижения ошибки (error performance), мудро комбинируя два эффекта.
   - Уменьшение уровня ошибки, вводимой порогом, при помощи целенаправленного избыточного обучения отдельных экспертов ассоциативной машины.
   - Уменьшение уровня ошибки, создаваемой дисперсией, за счет использования различных начальных состояний при обучении различных экспертов, и последующего усреднения по ансамблю их выходных сигналов.
2. Усиление улучшает эффективность алгоритма собственным оригинальным способом. В данном случае от отдельных экспертов требуется качество, лишь слегка отличающееся в лучшую сторону от случайного выбора. Слабая обучаемость экспертов преобразуется в сильную, при этом ошибку ассоциативной машины можно сделать сколь угодно малой. Эта выдающаяся метаморфоза достигается за счет *фильтрации* распределения входных данных, приводящей к тому, что слабо обучаемые модели (т.е. эксперты) в конечном итоге обучаются на всем распределении, либо за счет создания *подвыборки* множества обучения в соответствии с некоторым распределением, как в алгоритме AdaBoost. Преимущество последнего по сравнению с фильтрацией состоит в том, что он работает с обучающим множеством фиксированного размера.

Задачи

Усреднение по ансамблю

7.1. Рассмотрим ассоциативную машину, состоящую из K экспертов. Функция отображения входа на выход k-го эксперта обозначается через $F_k(\mathbf{x})$, где $\mathbf{x}$ — входной вектор, а $k = 1, 2, \ldots, K$. Выходы отдельных экспертов линейно суммируются, формируя общий выход системы y, следующим образом:

$$y = \sum_{k=1}^{K} w_k F_k(\mathbf{x}),$$

где w_k — линейный вес, назначенный $F_k(\mathbf{x})$. Требуется оценить w_k так, чтобы выход y обеспечивал оценку желаемого отклика d, соответствующего $\mathbf{x}$ по методу наименьших квадратов. Имея множество данных обучения $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, определите искомые значения w_k для решения этой задачи оценки параметров.

Усиление

7.2. Сравните вычислительные преимущества и недостатки методов усиления за счет фильтрации и AdaBoost.

7.3. Обычно усиление лучше всего выполнять на слабых моделях обучения, т.е. на моделях с относительно низким уровнем ошибок обобщения. Однако предположим, что имеется некоторая сильная модель обучения, т.е. модель с относительно высоким уровнем ошибок обобщения. Предположим также, что имеется множество примеров фиксированного размера. Как в такой ситуации выполнить усиление за счет фильтрации или алгоритм AdaBoost?

Смешение мнений экспертов

7.4. Рассмотрим следующую кусочно-линейную задачу:

$$F(x_1, x_2, \ldots, x_{10}) = \begin{cases} 3x_2 + 2x_3 + x_4 + 3 + \varepsilon, & \text{если } x_1 = 1, \\ 3x_5 + 2x_6 + x_7 - 3 + \varepsilon, & \text{если } x_1 = -1. \end{cases}$$

Для сравнения используются следующие конфигурации сети.

1. Многослойный персептрон: сеть $10 \rightarrow 10 \rightarrow 1$.
2. Смешение мнений экспертов: сеть шлюза: $10 \rightarrow 2$. Сеть эксперта: $10 \rightarrow 1$.

Сравните вычислительную сложность этих двух сетей.

7.5. Модель ME, описываемая функцией плотности условной вероятности (7.30), основана на модели скалярной регрессии, в которой ошибка имеет гауссово распределение со средним значением нуль и единичной дисперсией.

а) Переформулируйте это уравнение для более общего случая модели ME, соответствующей модели множественной регрессии, в которой желаемый отклик является вектором размерности q, а ошибка — многомерным гауссовым распределением с нулевым средним значением и матрицей ковариации σ.

б) Чем эта переформулированная модель отличается от модели ME, показанной на рис. 7.8?

7.6. Выведите алгоритм стохастического градиента для обучения модели смеси экспертов.

Иерархическое смешение мнений экспертов

7.7. а) Постройте блочную диаграмму модели HME с тремя уровнями иерархии. Предполагается, что для данной модели используется двоичное дерево решений.

б) Запишите апостериорные вероятности для нетерминальных узлов модели, описанной в пункте а). Продемонстрируйте рекурсивность вычислений, проводимых при оценке этих вероятностей.

в) Определите функцию плотности условной вероятности для модели HME, описанной в пункте а).

7.8. Обсудите сходства и отличия между моделью HME и сетями на основе радиально-базисных функций.

7.9. Выведите уравнения, описывающие алгоритм стохастического градиента для обучения модели HME с двумя уровнями иерархии. Предполагается, что для этой модели используется двоичное дерево решений.

Алгоритм EM и его применение в модели HME

7.10. Докажите свойство монотонного возрастания алгоритма EM, описываемого уравнением (7.62). Для этого выполните следующее.

а) Пусть

$$k(\mathbf{r}|d,\boldsymbol{\theta}) = \frac{f_c(\mathbf{r}|\boldsymbol{\theta})}{f_D(d|\boldsymbol{\theta})}$$

обозначает функцию плотности условной вероятности расширенного вектора параметров $\mathbf{r}$, для данного наблюдения d и вектора параметров $\boldsymbol{\theta}$. Исходя из этого, функция логарифмического правдоподобия на неполных данных будет иметь следующий вид:

$$L(\boldsymbol{\theta}) = L_c(\boldsymbol{\theta}) - \log k(\boldsymbol{r}|d, \theta),$$

где $L_c(\theta) = \log f_c(\mathbf{r}|\theta)$ — функция логарифмического правдоподобия на полных данных. Зная математическое ожидание функции $L(\theta)$ относительно условного распределения $\mathbf{r}$ и значение d, покажите, что

$$L(\boldsymbol{\theta}) = Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) - K(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)),$$

где

$$K(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) = E[\log k(\mathbf{r}|d, \hat{\boldsymbol{\theta}})].$$

При этих условиях покажите, что

$$L(\hat{\boldsymbol{\theta}}(n+1)) - L(\hat{\boldsymbol{\theta}}(n)) = \Big[Q(\hat{\boldsymbol{\theta}}(n+1), \hat{\boldsymbol{\theta}}(n)) - Q(\hat{\boldsymbol{\theta}}(n), \hat{\boldsymbol{\theta}}(n))\Big] - \\ - \Big[K(\hat{\boldsymbol{\theta}}(n+1), \hat{\boldsymbol{\theta}}(n)) - K(\hat{\boldsymbol{\theta}}(n), \hat{\boldsymbol{\theta}}(n))\Big].$$

б) *Неравенство Йенсена* (Jensen's inequality) утверждает, что если функция $g(\cdot)$ является выпуклой, а u — некоторая случайная переменная, то

$$E[g(u)] \leq g(E[u]),$$

где E — оператор математического ожидания. Более того, если $g(\cdot)$ — строго выпуклая функция, то из равенства в этом соотношении с вероятностью 1 следует, что $u = E(u)$ [221].

Используя неравенство Йенсена, покажите, что

$$K(\hat{\boldsymbol{\theta}}(n+1), \hat{\boldsymbol{\theta}}(n)) - K(\hat{\boldsymbol{\theta}}(n), \hat{\boldsymbol{\theta}}(n)) \leq 0. \tag{7.78}$$

Исходя из этого, покажите, что уравнение (7.62) выполняется для $n = 0, 1, 2, \ldots$.

7.11. Алгоритм EM довольно легко модифицируется для включения максимума апостериорной оценки вектора параметров $\boldsymbol{\theta}$. Используя правило Байеса, модифицируйте шаги E и M этого алгоритма, чтобы обеспечить эту оценку.

7.12. Если модель HME, обучаемая с помощью алгоритма EM, и многослойный персептрон, обучаемый по алгоритму обратного распространения, имеют одинаковые показатели производительности для данной задачи, интуитивно ожидается, что вычислительная сложность первого будет превосходить сложность второго. Приведите аргументы за или против этого утверждения.

7.13. Обоснуйте связь между переменными-индикаторами и соответствующими апостериорными вероятностями, описанными в формулах (7.66) и (7.68).

7.14. Уравнение (7.75) описывает взвешенный метод наименьших квадратов для оптимизации сетей экспертов модели HME, представленной на рис. 7.11, в предположении, что желаемый отклик d — скаляр. Как модифицировать это соотношение для случая многомерного желаемого отклика?

Анализ главных компонентов

8.1. Введение

Важным свойством нейронных сетей является их способность обучаться на основе примеров из окружающей среды и с помощью этого обучения повышать производительность работы. В предыдущих четырех главах основное внимание фокусировалось на алгоритмах обучения с учителем, в которых множество целей определяется внешним учителем. Целью обучения является построение желаемого отображения входного сигнала в выходной, которое сеть должна аппроксимировать. В этой и последующих трех главах рассматриваются алгоритмы *самоорганизующегося обучения* (self-organized learning), или *обучения без учителя* (unsupervised learning). Целью алгоритмов самоорганизующегося обучения является *выявление* (discover) в множестве входных данных существенных образов или признаков, причем этот процесс проходит без участия учителя. Для этого алгоритм реализует множество правил *локальной* природы, что позволяет обучаться вычислению отображения входного сигнала на выходной с требуемыми свойствами. Здесь термин "локальный" подразумевает следующее. Изменения синаптических весов нейрона определяются только непосредственными соседями этого нейрона. Модели сетей, обучаемые на основе принципа самоорганизации, в гораздо большей мере отражают свойства нейробиологических структур, нежели архитектуры, обучаемые с учителем. Это неудивительно, поскольку подобный принцип организации сетей отражает принципы функционирования мозга.

Архитектура самоорганизующихся систем может принимать множество совершенно различных форм. Например, такая сеть может состоять из *входного* (input layer) и *выходного* слоя (output layer), связанных прямыми связями, и включать латеральные связи между нейронами второго слоя. Еще одним примером могут служить многослойные сети прямого распространения, в которых самоорганизация проявляется при переходе от одного слоя к другому. В обоих случаях процесс обучения состоит в периодически повторяющемся изменении синаптических весов всех связей в системе в ответ на подачу входных образов в соответствии с предписанными правилами до получения конечной конфигурации системы.

Эта глава посвящена системам самоорганизации, основанным на принципе обучения Хебба. Основное внимание в ней уделяется *анализу главных компонентов* (principal components analysis) — стандартному приему, обычно используемому для уменьшения размерности данных в статистических системах распознавания образов и обработки сигналов.

Структура главы

Материал этой главы организован следующим образом. В разделе 8.2 на качественном уровне описываются основные принципы самоорганизации. В разделе 8.3 представлен вводный материал по анализу главных компонентов, который будет положен в основу последующего исследования самоорганизующихся систем в настоящей главе.

Вооружившись этими фундаментальными знаниями, мы приступим к изучению отдельных самоорганизующихся систем. В разделе 8.4 описывается простая модель, состоящая из единственного нейрона и позволяющая извлечь первый главный компонент на основе самоорганизации. В разделе 8.5 рассматривается более сложная самоорганизующаяся система в форме сети прямого распространения с одним слоем нейронов, которая извлекает все главные компоненты, основываясь на работе предыдущей простой модели. Эта процедура будет проиллюстрирована компьютерным экспериментом по кодированию изображений, представленным в разделе 8.6. В разделе 8.7 вниманию читателя будет предложена еще одна самоорганизующаяся система, выполняющая аналогичную функцию. Однако эта система является еще более сложной, так как содержит латеральные связи. В разделе 8.8 приводится классификация алгоритмов анализа главных компонентов на основе нейронных сетей. Раздел 8.9 посвящен классификации алгоритмов снижения размерности данных на адаптивные и пакетные методы.

В разделе 8.10 описывается нелинейная форма анализа главных компонентов, основанная на идее ядра скалярного произведения, определенного в соответствии с теоремой Мерсера, о которой речь шла в главе 6, посвященной машинам опорных векторов.

Завершается глава заключительными размышлениями об анализе главных компонентов.

8.2. Некоторые интуитивные принципы самоорганизации

Как уже говорилось ранее, обучение на основе самоорганизации (без учителя) заключается в последовательном изменении синаптических весов нейронной сети в ответ на возбуждающие сигналы, производимые в соответствии с заранее определенными правилами, повторяющемся до тех пор, пока не будет сформирована окончательная

конфигурация системы. Ключевой вопрос, естественно, заключается в том, как с помощью самоорганизации сформировать эту окончательную структуру. Ответ на него основывается на следующем наблюдении [1061].

Глобальный порядок определяется локальными взаимодействиями.

Это наблюдение имеет первостепенную важность. Его можно применить и к мозгу, и к искусственным нейронным сетям. В частности, многие изначально случайные взаимодействия соседних нейронов сети могут перерасти в состояние глобального порядка и в конечном счете привести к согласованному поведению в форме пространственных моделей или временных ритмов. Это и является сущностью самоорганизации.

Организация сетей формируется на двух различных уровнях, которые взаимодействуют друг с другом с помощью *обратной связи*. Этими двумя уровнями являются следующие.

- *Уровень активности* (activity). В ответ на входные возмущения данная сеть формирует определенные образы (последовательность действий).
- *Уровень связности* (connectivity). Связи (синаптические веса) сети изменяются в ответ на нейронные сигналы образов активности благодаря синаптической пластичности.

Для того чтобы сеть достигла самоорганизации (а не стабилизации), обратная связь между изменениями в синаптических весах и изменениями в образах активности должна быть *положительной* (positive). В соответствии с этим можно вывести первый принцип самоорганизации [1097].

ПРИНЦИП 1

Изменение синаптических весов ведет к самоусилению сети (self-amplify).

Процесс самоусиления ограничен требованием того, чтобы изменения синаптических весов основывались на локально доступных сигналах, а именно на предсинаптических и постсинаптических. Это требование усиления и локальности определяет механизм, в котором сильные синапсы обеспечивают согласование предсинаптических сигналов с постсинаптическими. В свою очередь, синапс усиливается за счет такого согласования. Описанный здесь механизм на самом деле является подтверждением постулата обучения Хебба.

Для стабилизации системы должна существовать некоторая форма конкуренции за "ограниченные" ресурсы (количество входных сигналов или энергию). В частности, усиление отдельных синапсов сети должно компенсироваться ослаблением остальных. Следовательно, усиливаться могут только "успешные" синапсы, в то время как "менее удачливые" имеют тенденцию к ослаблению и постепен-

ному исчезновению. Это наблюдение приводит к формулировке второго принципа самоорганизации [1097].

ПРИНЦИП 2

Ограниченность ресурсов ведет к конкуренции между синапсами и, таким образом, к выбору наиболее успешно развивающихся синапсов за счет других (т.е. наиболее подходящих).

Этот принцип реализуется благодаря пластичности синапсов.

Для следующего наблюдения заметим, что отдельный синапс не может эффективно реализовать благоприятные события. Для этого необходима совместная работа группы синапсов, расположенных на входе отдельного нейрона, и достаточное усиление согласованных сигналов для его активации. Таким образом, можно сформулировать еще один принцип самоорганизации [1097].

ПРИНЦИП 3

Модификация синаптических весов ведет к кооперации.

Присутствие сильного синапса может усилить и другие синапсы в свете общей системы конкуренции в сети. Эта форма кооперации проистекает из пластичности синапсов или из одновременного стимулирования предсинаптических нейронов, вызванного наличием благоприятных условий во внешней среде.

Все три принципа самоорганизации относятся только к нейронным сетям. Однако, для того чтобы самоорганизующееся обучение обеспечивало полезную функцию обработки информации, в образах активации, передаваемых нейронам из внешней среды, должна существовать доля *избыточности* (redundancy). Вопрос избыточности будет обсуждаться при рассмотрении теории информации Шеннона в главе 10. Здесь же мы ограничимся формулировкой последнего принципа самоорганизующегося обучения [90].

ПРИНЦИП 4

Порядок и структура образов активации содержат избыточную информацию, накапливаемую сетью в форме знаний, являющихся необходимым условием самоорганизующегося обучения.

Некоторая часть знаний может быть получена с помощью наблюдения за статистическими параметрами, такими как среднее значение, дисперсия и матрица корреляции входных данных.

Принципы 1–4 самоорганизующегося обучения образуют нейробиологический базис для адаптивных алгоритмов анализа главных компонентов, которые описываются в настоящей главе, а также для самоорганизующихся карт Кохонена (Kohonen), которые рассматриваются в главе 9. Эти принципы реализованы также и в ряде других

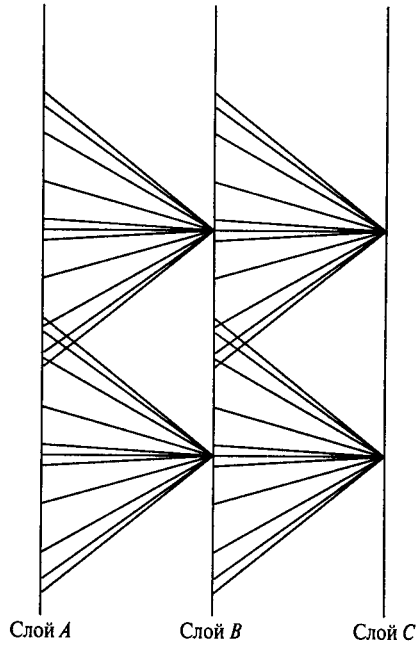


Рис. 8.1. Структура модульной самонастраивающейся сети

самоорганизующихся моделей, которые основаны на знаниях из области нейробиологии. Одной из таких моделей, которую нельзя обойти вниманием, является *модель Линскера* (Linsker's model) зрительной системы млекопитающего [656].

Анализ признаков на основе самоорганизации

Обработка сигналов в системах зрения производится поэтапно. В частности, простые признаки, такие как контрастность или ориентация контуров, анализируются на ранних этапах, в то время как более сложные признаки — на поздних. На рис. 8.1 показана обобщенная структура модульной сети, которая имитирует зрительную систему. В модели Линскера нейроны сети (см. рис. 8.1) организованы в двумерные слои с прямыми переходами от одного слоя к следующему. Все нейроны получают информацию от ограниченного числа нейронов, расположенных в пересекающихся областях предыдущего слоя, составляющих *поле восприятия*, или *рецепторное поле* (receptive field), нейрона. Поля восприятия сети играют ключевую роль в процессе построения синаптических связей, так как они обеспечивают нейронам одного слоя возможность реагировать на *пространственные корреляции* активности нейронов предыдущего слоя. Нужно сделать два структурных допущения.

1. После выбора положения синаптических связей они остаются фиксированными на протяжении всего процесса развития нейронной структуры.
2. Все нейроны выступают в качестве линейных сумматоров.

Эта модель содержит черты Хеббовского принципа модификации синапсов, а также конкурентного способа обучения. Таким образом, выходы сети оптимально выделяются среди ансамбля входов, при этом самоорганизующееся обучение продвигается от слоя к слою. Это значит, что в процессе обучения на основе самоорганизации для каждого слоя выполняется анализ признаков и создается собственная структура. В [656] были представлены результаты моделирования, которые качественно подтвердили свойства, обнаруженные на ранних стадиях обработки визуальных сигналов в зрительных системах котов и обезьян. Признавая в высочайшей степени сложную природу зрительных систем, следует отметить, что простая модель, рассмотренная Линскером, оказалась способной сконструировать аналогичные нейроны для анализа признаков. Это совсем не говорит о том, что нейроны-анализаторы признаков зрительной системы млекопитающих имеют точно такую же природу, которая описывается моделью Линскера. Более того, такие структуры могут создаваться относительно простыми многослойными сетями, в которых синаптические связи формируются согласно Хеббовской форме обучения.

Тем не менее в этой главе главный интерес для нас будут представлять анализ главных компонентов и способы его выполнения с помощью самоорганизующихся систем, основанных на принципе обучения Хебба.

8.3. Анализ главных компонентов

Главной задачей в статистическом распознавании является *выделение признаков* (feature selection) или *извлечение признаков* (feature extraction). Под выделением признаков понимается процесс, в котором *пространство данных* (data space) преобразуется в *пространство признаков* (feature space), теоретически имеющее ту же размерность, что и исходное пространство. Однако обычно преобразования выполняются таким образом, чтобы пространство данных могло быть представлено сокращенным количеством "эффективных" признаков. Таким образом, остается только существенная часть информации, содержащейся в данных. Другими словами, множество данных подвергается *сокращению размерности* (dimensionality reduction). Для большей конкретизации предположим, что существует некоторый вектор $\mathbf{x}$ размерности m, который мы хотим передать с помощью l чисел, где $l < m$. Если мы просто обрежем вектор $\mathbf{x}$, это приведет к тому, что среднеквадратическая ошибка будет равна сумме дисперсий элементов, "вырезанных" из вектора $\mathbf{x}$. Поэтому возникает вопрос: "Существует ли такое обратимое *линейное* преобразование $\mathbf{T}$, для которого обрезание вектора $\mathbf{Tx}$ будет оптимальным в смысле среднеквадратической ошибки?" Естественно, при этом

преобразование $\mathbf{T}$ должно иметь свойство маленькой дисперсии своих отдельных компонентов. *Анализ главных компонентов* (в теории информации он называется *преобразование Карунена–Лоева* (*Karhunen-Loeve transformation*)) максимизирует скорость уменьшения дисперсии и, таким образом, вероятность правильного выбора. В этой главе описываются алгоритмы обучения, основанные на принципах Хебба, которые осуществляют анализ главных компонентов[1] интересующего вектора данных.

Пусть $\mathbf{X}$ — m-мерный *случайный вектор*, представляющий интересующую нас среду. Предполагается, что он имеет нулевое среднее значение

$$E(\mathbf{X}) = \mathbf{0},$$

где E — оператор статистического ожидания. Если $\mathbf{X}$ имеет ненулевое среднее, можно вычесть это значение еще до начала анализа. Пусть $\mathbf{q}$ — *единичный вектор* (unit vector) размерности m, на который *проектируется* вектор $\mathbf{X}$. Эта проекция определяется как скалярное произведение векторов $\mathbf{X}$ и $\mathbf{q}$:

$$A = \mathbf{X}^T\mathbf{q} = \mathbf{q}^T\mathbf{X} \tag{8.1}$$

при ограничении

$$||\mathbf{q}|| = (\mathbf{q}^T\mathbf{q})^{1/2} = 1. \tag{8.2}$$

Проекция A представляет собой случайную переменную со средним значением и с дисперсией, связанными со статистикой случайного вектора $\mathbf{X}$. В предположении, что случайный вектор $\mathbf{X}$ имеет нулевое среднее значение, среднее значение его проекции A также будет нулевым:

$$E[A] = \mathbf{q}^T E[\mathbf{X}] = 0.$$

Таким образом, дисперсия A равна

$$\sigma^2 = E[A^2] = E[(\mathbf{q}^T\mathbf{X})(\mathbf{X}^T\mathbf{q})] = \mathbf{q}^T E[\mathbf{X}\mathbf{X}^T]\mathbf{q} = \mathbf{q}^T\mathbf{R}\mathbf{q}. \tag{8.3}$$

Матрица $\mathbf{R}$ размерности $m \times m$ является *матрицей корреляции* случайного вектора $\mathbf{X}$, определяемой как ожидание произведения случайного вектора $\mathbf{X}$ самого на себя:

---

[1] Анализ главных компонентов (Principal Component Analysis – PCA) является, пожалуй, самым старым и известным методом многомерного анализа [517], [857]. Впервые он использовался в [824] в биологическом контексте для придания анализу линейной регрессии новой формы. Эта идея была впоследствии развита в [487], посвященной психометрии. Совершенно независимо от этой работы эта идея появилась при выводе теории вероятности в [541] и впоследствии была обобщена в [671].

$$\mathbf{R} = E[\mathbf{X}\mathbf{X}^T]. \tag{8.4}$$

Матрица $\mathbf{R}$ является *симметричной*, т.е.

$$\mathbf{R}^T = \mathbf{R}. \tag{8.5}$$

Из этого следует, что если $\mathbf{a}$ и $\mathbf{b}$ — произвольные векторы размерности $m \times 1$, то

$$\mathbf{a}^T\mathbf{R}\mathbf{b} = \mathbf{b}^T\mathbf{R}\mathbf{a}. \tag{8.6}$$

Из выражения (8.3) видно, что дисперсия σ^2 проекции A является функцией единичного вектора $\mathbf{q}$. Таким образом, можно записать:

$$\psi(q) = \sigma^2 = \mathbf{q}^T\mathbf{R}\mathbf{q}, \tag{8.7}$$

на основании чего $\psi(\mathbf{q})$ можно представить как *дисперсионный зонд* (variance probe).

Структура анализа главных компонентов

Следующим вопросом, подлежащим рассмотрению, является поиск тех единичных векторов $\mathbf{q}$, для которых функция $\psi(\mathbf{q})$ имеет *экстремальные* или *стационарные* значения (локальные максимумы и минимумы) при ограниченной Евклидовой норме вектора $\mathbf{q}$. Решение этой задачи лежит в собственной структуре матрицы корреляции $\mathbf{R}$. Если $\mathbf{q}$ — единичный вектор, такой, что дисперсионный зонд $\psi(\mathbf{q})$ имеет экстремальное значение, то для любого возмущения $\delta\mathbf{q}$ единичного вектора $\mathbf{q}$ выполняется

$$\psi(\mathbf{q} + \delta\mathbf{q}) = \psi(\mathbf{q}). \tag{8.8}$$

Из определения дисперсионного зонда можем вывести следующее соотношение:

$$\psi(\mathbf{q} + \delta\mathbf{q}) = (\mathbf{q} + \delta\mathbf{q})^T\mathbf{R}(\mathbf{q} + \delta\mathbf{q}) = \mathbf{q}^T\mathbf{R}\mathbf{q} + 2(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} + (\delta\mathbf{q})^T\mathbf{R}\delta\mathbf{q},$$

где во второй строке использовалось выражение (8.6). Игнорируя слагаемое второго порядка $(\delta\mathbf{q})^T\mathbf{R}\delta\mathbf{q}$ и используя определение (8.7), можно записать следующее:

$$\psi(\mathbf{q} + \delta\mathbf{q}) = \mathbf{q}^T\mathbf{R}\mathbf{q} + 2(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} = \psi(\mathbf{q}) + 2(\delta\mathbf{q})^T\mathbf{R}\mathbf{q}. \tag{8.9}$$

Отсюда, подставляя (8.8) в (8.9), получим:

$$2(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} = 0. \tag{8.10}$$

Естественно, любые возмущения $\delta\mathbf{q}$ вектора $\mathbf{q}$ нежелательны; ограничим их только теми возмущениями, для которых норма возмущенного вектора $\mathbf{q}+\delta\mathbf{q}$ остается равной единице, т.е.

$$||\mathbf{q} + \delta\mathbf{q}|| = 1$$

или, что эквивалентно,

$$(\mathbf{q} + \delta\mathbf{q})^T(\mathbf{q} + \delta\mathbf{q}) = 1.$$

Исходя из этого, в свете равенства (8.2) требуется, чтобы для возмущения первого порядка $\delta\mathbf{q}$ выполнялось соотношение

$$(\delta\mathbf{q})^T\mathbf{q} = 0. \tag{8.11}$$

Это значит, что возмущения $\delta\mathbf{q}$ должны быть ортогональны вектору $\mathbf{q}$ и, таким образом, допускаются только изменения в направлении вектора $\mathbf{q}$.

Согласно соглашению, элементы единичного вектора $\mathbf{q}$ являются безразмерными в физическом смысле. Таким образом, можно скомбинировать (8.10) и (8.11), введя дополнительный масштабирующий множитель λ в последнее равенство с той же размерностью, что и вхождение в матрицу корреляции $\mathbf{R}$. После этого можно записать следующее:

$$(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} - \lambda(\delta\mathbf{q})^T\mathbf{q} = 0,$$

или, эквивалентно,

$$(\delta\mathbf{q})^T(\mathbf{R}\mathbf{q} - \lambda\mathbf{q}) = 0. \tag{8.12}$$

Для того чтобы выполнялось условие (8.12), необходимо и достаточно, чтобы

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q}. \tag{8.13}$$

Это — уравнение определения таких единичных векторов $\mathbf{q}$, для которых дисперсионный зонд $\psi(\mathbf{q})$ принимает экстремальные значения.

В уравнении (8.13) можно легко узнать *задачу определения собственных значений* (eigenvalue: problem) из области линейной алгебры [1022]. Эта задача имеет нетривиальные решения (т.е. $\mathbf{q} \neq \mathbf{0}$) только для некоторых значений λ, которые и называются *собственными значениями* (eigenvalue) матрицы корреляции $\mathbf{R}$. При этом соответствующие векторы $\mathbf{q}$ называют *собственными векторами* (eigenvector). Матрица корреляции характеризуется действительными, неотрицательными собственными значениями. Соответствующие собственные векторы являются единичными (если все собственные значения различны). Обозначим собственные значения матрицы $\mathbf{R}$ размерности $m \times m$ как $\lambda_1, \lambda_2, \ldots, \lambda_m$, а соответствующие им собственные векторы – $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m$ соответственно. Тогда можно записать следующее:

$$\mathbf{R}\mathbf{q}_j = \lambda_j \mathbf{q}_j, j = 1, 2, \ldots, m. \tag{8.14}$$

Пусть соответствующие собственные значения упорядочены следующим образом:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_j > \ldots > \lambda_m. \tag{8.15}$$

При этом λ_1 будет равно $\lambda_{\max}$. Пусть из соответствующих собственных векторов построена следующая матрица размерности $m \times m$:

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_j, \ldots, \mathbf{q}_m]. \tag{8.16}$$

Тогда систему m уравнений (8.14) можно объединить в одно матричное уравнение:

$$\mathbf{R}\mathbf{Q} = \mathbf{Q}\lambda, \tag{8.17}$$

где $\mathbf{\Lambda}$ – диагональная матрица, состоящая из собственных значений матрицы $\mathbf{R}$:

$$\mathbf{\Lambda} = \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_j, \ldots, \lambda_m]. \tag{8.18}$$

Матрица $\mathbf{Q}$ является *ортогональной (унитарной)* в том смысле, что векторы-столбцы (т.е. собственные векторы матрицы $\mathbf{R}$) удовлетворяют условию ортогональности:

$$\mathbf{q}_i^T \mathbf{q}_j = \begin{cases} 1, & j = i, \\ 0, & j \neq i. \end{cases} \tag{8.19}$$

Выражение (8.19) предполагает, что собственные значения различны. Эквивалентно, можно записать:

$$\mathbf{Q}^T \mathbf{Q} = \mathbf{I},$$

из чего можно заключить, что обращение матрицы $\mathbf{Q}$ эквивалентно ее транспонированию:

$$\mathbf{Q}^T = \mathbf{Q}^{-1}. \tag{8.20}$$

Это значит, что выражение (8.17) можно переписать в форме, называемой *ортогональным преобразованием подобия* (orthogonal similarity transformation):

$$\mathbf{Q}^T\mathbf{R}\mathbf{Q} = \mathbf{\Lambda}, \tag{8.21}$$

или в расширенной форме:

$$\mathbf{q}_j^T\mathbf{R}\mathbf{q}_k = \begin{cases} \lambda_j, & k = j, \\ 0, & k \neq j. \end{cases} \tag{8.22}$$

Ортогональное преобразование подобия (8.21) трансформирует матрицу корреляции $\mathbf{R}$ в диагональную матрицу, состоящую из собственных значений. Сама матрица корреляции может быть выражена в терминах своих собственных векторов и собственных значений следующим образом:

$$\mathbf{R} = \sum_{i=1}^{m} \lambda_i \mathbf{q}_i \mathbf{q}_i^T. \tag{8.23}$$

Это выражение называют *спектральной теоремой* (spectral theorem). Произведение векторов $\mathbf{q}_i\mathbf{q}_i^T$ имеет ранг 1 для всех i.

Уравнения (8.21) и (8.23) являются двумя эквивалентными представлениями *разложения по собственным векторам* (eigencomposition) матрицы корреляции $\mathbf{R}$.

Анализ главных компонентов и разложение по собственным векторам матрицы $\mathbf{R}$ являются в сущности одним и тем же; различается только подход к задаче. Эта эквивалентность следует из уравнений (8.7) и (8.23), из которых ясно видно равенство собственных значений и дисперсионного зонда, т.е.

$$\psi(\mathbf{q}_j) = \lambda_j, j = 1, 2, \ldots, m. \tag{8.24}$$

Теперь можно сделать выводы, касающиеся анализа главных компонентов.

- Собственные векторы матрицы корреляции $\mathbf{R}$ принадлежат случайному вектору $\mathbf{X}$ с нулевым средним значением и определяют единичные векторы $\mathbf{q}_j$, представляющие основные направления, вдоль которых дисперсионный зонд $\psi(\mathbf{q}_j)$ принимает экстремальные значения.
- Соответствующие собственные значения определяют экстремальные значения дисперсионного зонда $\psi(\mathbf{u}_j)$.

Основные представления данных

Пусть *вектор данных* $\mathbf{x}$ является реализацией случайного вектора $\mathbf{X}$.

При наличии m возможных значений единичного вектора $\mathbf{q}$ следует рассмотреть m возможных проекций вектора данных $\mathbf{x}$. В частности, согласно формуле (8.1)

$$a_j = \mathbf{q}_j^T \mathbf{x} = \mathbf{x}^T \mathbf{q}_j, \quad j = 1, 2, \ldots, m, \tag{8.25}$$

где a_j — проекции вектора $\mathbf{x}$ на основные направления, представленные единичными векторами $\mathbf{u}_j$. Эти проекции a_j называют *главными компонентами* (principal component). Их количество соответствует размерности вектора данных $\mathbf{x}$. При этом формулу (8.25) можно рассматривать как процедуру *анализа* (analysis).

Для того чтобы восстановить вектор исходных данных $\mathbf{x}$ непосредственно из проекций a_j, выполним следующее. Прежде всего объединим множество проекций $\{a_j | j = 1, 2, \ldots, m\}$ в единый вектор:

$$\mathbf{a} = [a_1, a_2, \ldots, a_m]^T = [\mathbf{x}^T \mathbf{q}_1, \mathbf{x}^T \mathbf{q}_2, \ldots, \mathbf{x}^T \mathbf{q}_m]^T = \mathbf{Q}^T \mathbf{x}. \tag{8.26}$$

Затем перемножим обе части уравнения (8.26) на матрицу $\mathbf{Q}$, после чего используем соотношение (8.20). В результате исходный вектор данных $\mathbf{x}$ будет реконструирован в следующем виде:

$$\mathbf{x} = \mathbf{Q}\mathbf{a} = \sum_{j=1}^{m} a_j \mathbf{q}_j, \tag{8.27}$$

который можно рассматривать как формулу *синтеза*. В этом контексте единичные векторы $\mathbf{q}_j$ будут представлять собой пространства данных. И в самом деле, выражение (8.27) является не чем иным, как преобразованием координат, в соответствии с которым точки $\mathbf{x}$ пространства данных преобразуются в соответствующие точки $\mathbf{a}$ пространства признаков.

Сокращение размерности

С точки зрения задачи статистического распознавания практическое значение анализа главных компонентов состоит в том, что он обеспечивает эффективный способ *сокращения размерности* (dimensionality reduction). В частности, можно сократить количество признаков, необходимых для эффективного представления данных, устраняя те линейные комбинации в (8.27), которые имеют малые дисперсии, и оставляя те, дисперсии которых велики. Пусть $\lambda_1, \lambda_2, \ldots, \lambda_l$ — наибольшие l собственных значений матрицы корреляции $\mathbf{R}$. Тогда вектор данных $\mathbf{x}$ можно аппроксимировать, отсекая члены разложение (8.27) после l-го слагаемого:

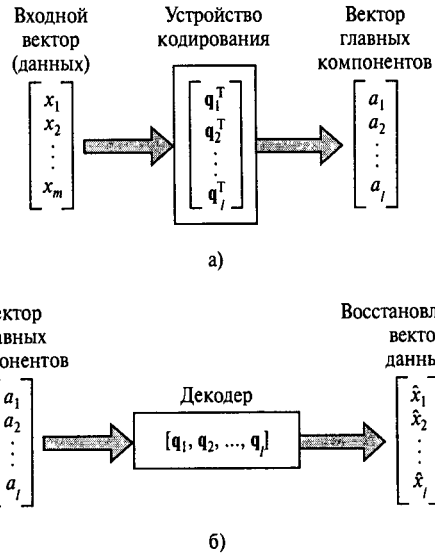


Рис. 8.2. Два этапа анализа главных компонентов: кодирование (а) и декодирование (б)

$$\hat{\mathbf{x}} = \sum_{j=1}^{l} a_j \mathbf{q}_j = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_l] \begin{bmatrix} a_1 \\ a_2 \\ \ldots \\ a_l \end{bmatrix}, \quad l \leq m. \tag{8.28}$$

Имея исходный вектор $\mathbf{x}$, с помощью выражения (8.25) можно вычислить главные компоненты из (8.28) следующим образом:

$$\begin{bmatrix} a_1 \\ a_2 \\ \ldots \\ a_l \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \ldots \\ \mathbf{q}_l^T \end{bmatrix} \mathbf{x}, \quad l \leq m. \tag{8.29}$$

Линейная проекция (8.29) из $\Re^m$ в $\Re^l$ (т.е. отображение из пространства данных в пространство признаков) представляет собой *шифратор* (encoder) для приближенного представления вектора данных $\mathbf{x}$ (рис. 8.2, *а*). Соответственно линейная проекция (8.28) из $\Re^l$ в $\Re^m$ (т.е. обратное отображение пространства признаков в пространство данных) представляет собой *дешифратор* (decoder) (см. рис. 8.2, *б*). Обратите внимание, что *доминирующие* (dominant) собственные значения $\lambda_1, \lambda_2, \ldots, \lambda_l$ не участвуют в вычислениях (8.28) и (8.29). Они просто определяют количество главных компонентов, используемых для кодирования и декодирования.

Вектор ошибки аппроксимации (approximation error vector) $\mathbf{e}$ равен разности между вектором исходных данных $\mathbf{x}$ и вектором приближенных данных $\hat{\mathbf{x}}$ (рис. 8.3):

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}. \tag{8.30}$$

Подставляя (8.27) и (8.28) в (8.30), получим:

$$\mathbf{e} = \sum_{j=l+1}^{m} a_j \mathbf{q}_j. \tag{8.31}$$

Вектор ошибки $\mathbf{e}$ является ортогональным вектору приближенных данных $\hat{\mathbf{x}}$ (рис. 8.3). Другими словами, скалярное произведение векторов $\mathbf{e}$ и $\hat{\mathbf{x}}$ равно нулю. Используя (8.28) и (8.31), это свойство можно доказать следующим образом:

$$\mathbf{e}^T \hat{\mathbf{x}} = \sum_{i=l+1}^{m} a_i \mathbf{q}_i^T \sum_{j=1}^{l} a_j \mathbf{q}_j = \sum_{i=l+1}^{m} \sum_{j=1}^{l} a_i a_j \mathbf{q}_i^T \mathbf{q}_j = 0, \tag{8.32}$$

где учитывается второе условие выражения (8.19). Соотношение (8.32) называют *принципом ортогональности* (principle of orthogonality).

Общая дисперсия m компонентов вектора данных $\mathbf{x}$ составляет (согласно (8.7) и (8.22)):

$$\sum_{j=1}^{m} \sigma_j^2 = \sum_{j=1}^{m} \lambda_j, \tag{8.33}$$

где σ_j^2 — дисперсия j-го главного компонента a_j. Общая дисперсия l элементов приближенного вектора $\hat{\mathbf{x}}$ равна:

$$\sum_{j=1}^{l} \sigma_j^2 = \sum_{j=1}^{l} \lambda_j. \tag{8.34}$$

Таким образом, общая дисперсия $(l - m)$ элементов вектора ошибки аппроксимации $\mathbf{x} - \hat{\mathbf{x}}$ равна:

$$\sum_{j=l+1}^{m} \sigma_j^2 = \sum_{j=l+1}^{m} \lambda_j. \tag{8.35}$$

Числа $\lambda_{l+1}, \ldots, \lambda_m$ являются наименьшими $(m - l)$ собственными значениями матрицы корреляции $\mathbf{R}$. Они соответствуют слагаемым, исключенным из разложения (8.28), используемого для построения приближенного вектора $\hat{\mathbf{x}}$.Чем ближе эти собственные значения к нулю, тем более эффективным будет сокращение размерности (как результат применения анализа главных компонентов вектора данных $\mathbf{x}$) в пред-

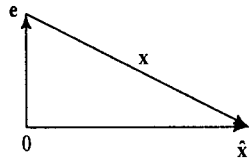


Рис. 8.3. Взаимосвязь вектора **x**, его реконструированной версии $\hat{\mathbf{x}}$ и вектора ошибки **e**

ставлении информации исходных данных. Таким образом, для того чтобы обеспечить сокращение размерности входных данных, нужно *вычислить собственные значения и собственные векторы матрицы корреляции векторов входных данных, а затем ортогонально проектировать эти данные на подпространство, задаваемое собственными векторами, соответствующими доминирующим собственным значениям этой матрицы*. Этот метод представления данных обычно называют *пространственной декомпозицией* (subspace decomposition) [797].

Пример 8.1

Двумерное множество данных

Для иллюстрации применения анализа главных компонентов рассмотрим пример двумерного множества данных (рис. 8.4). При этом предполагается, что обе оси признаков на графике имеют приблизительно одинаковый масштаб. Горизонтальная и вертикальная оси графика представляют собой естественные координаты множества данных. Повернутые оси с метками 1 и 2 явились результатом применения к этому множеству данных анализа главных компонентов. На рис. 8.4 видно, что проектирование множества данных на ось 1 позволяет выделить свойство выпуклости данных, а именно тот факт, что множество данных является бимодальным (т.е. в его структуре существуют два кластера). И в самом деле, дисперсия проекции точек данных на ось 1 превышает дисперсию проекции на любую другую ось рисунка. В отличие от этой ситуации внутренняя бимодальная природа этого множества данных абсолютно не видна при его проектировании на ортогональную ось 2.

В этом простом примере важно обратить внимание на то, что кластерная структура данного множества не проявляется даже при проектировании на вертикальную и горизонтальную оси графика. На практике чаще всего приходится иметь дело с многомерными множествами данных, в которых кластерная структура данных скрыта, и приходится выполнять статистический анализ, аналогичный описанному выше анализу главных компонентов [653]. ■

8.4. Фильтр Хебба для выделения максимальных собственных значений

Существует тесная взаимосвязь между поведением самоорганизующихся нейронных сетей и статистическим методом анализа главных компонентов. В этом разделе мы продемонстрируем эту связь, доказав следующее утверждение. Один линейный нейрон с Хеббовским правилом адаптации синаптических весов может

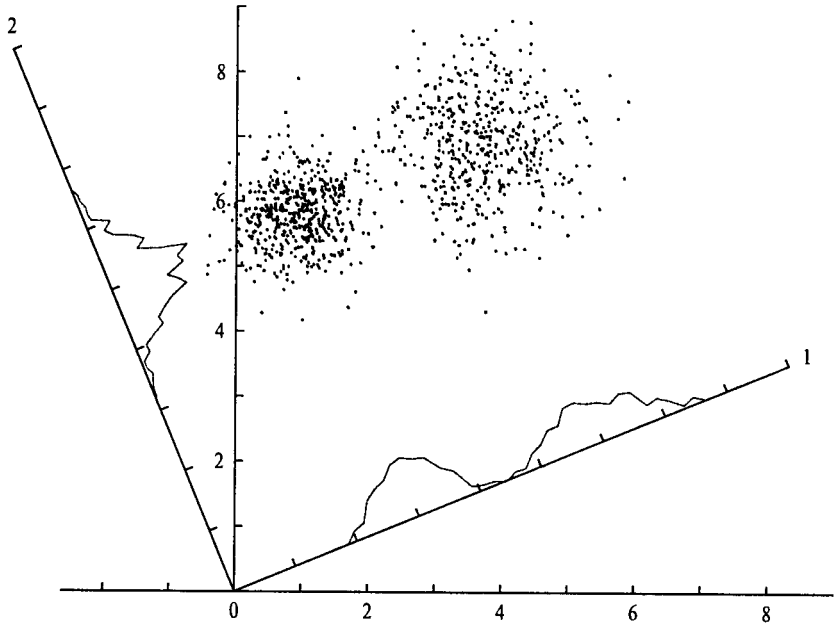


Рис. 8.4. Совокупность данных показана в двумерной системе координат, а плотности точек сформированы проектированием этих совокупностей на две оси, обозначенные цифрами 1 и 2. Проекция на ось 1 имеет максимальную дисперсию и явно демонстрирует бимодальный или кластерный характер данных

быть преобразован в фильтр для выделения первого главного компонента входного распределения [798].

Чтобы приступить к изучению этого вопроса, рассмотрим простую нейронную модель, показанную на рис. 8.5, *а*. Эта модель является *линейной* в том смысле, что ее выход является линейной комбинацией входов. Нейрон получает множество из m входных сигналов $x_1, x_2, \ldots, x_m$ через соответствующее множество m синапсов $w_1, w_2, \ldots, w_m$. Выход полученной модели можно определить как

$$y = \sum_{i=1}^{m} w_i x_i. \tag{8.36}$$

Обратите внимание, что в описанной здесь ситуации мы имеем дело с единственным нейроном, поэтому нет необходимости применять в обозначениях синаптических весов двойной индекс.

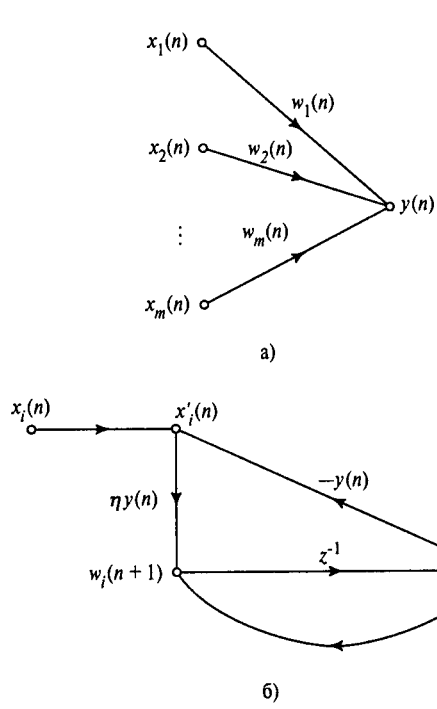


Рис. 8.5. Представление фильтра для извлечения максимального собственного значения в виде графа передачи сигнала: граф выражения (8.36) (а); граф выражений (8.41) и (8.42) (б)

Согласно постулату Хебба, синаптический вес w_i изменяется во времени, сильно возрастая, если предсинаптический сигнал x_i и постсинаптический сигнал y совпадают друг с другом. В частности, можно записать следующее:

$$w_i(n+1) = w_i(n) + \eta y(n) x_i(n), i = 1, 2, \ldots, m, \tag{8.37}$$

где n — дискретное время; η — *параметр интенсивности обучения* (learning-rate parameter). Однако это правило обучения в своей общей форме приводит к неограниченному росту синаптических весов w_i, что неприемлемо с физической точки зрения. Эту проблему можно обойти, применив в правиле обучения, используемом для адаптации синаптических весов, некоторую форму *насыщения* (saturation) или *нормализации* (normalization). Использование нормализации обеспечивает эффект введения конкуренции между синапсами нейрона за обладание ограниченными ресурсами, которая, исходя из второго принципа самоорганизации, является существенным условием стабилизации сети. С математической точки зрения удобная форма нормализации может быть описана следующим соотношением [798]:

$$w_i(n+1) = \frac{w_i(n) + \eta y(n) x_i(n)}{(\sum_{i=1}^{m} [w_i(n) + \eta y(n) x_i(n)]^2)^{1/2}}, \tag{8.38}$$

где суммирование в знаменателе проводится по всему множеству синапсов, связанных с данным нейроном. Предполагая малость параметра скорости обучения η, (8.38) можно представить в виде ряда по η и записать так:

$$w_i(n+1) = w_i(n) + \eta y(n)[x_i(n) - y(n)w_i(n)] + O(\eta^2), \tag{8.39}$$

где элемент $O(\eta^2)$ представляет собой слагаемые второго и более высоких порядков по η. Для малых значений η это слагаемое вполне обоснованно может быть проигнорировано. Таким образом, (8.38) можно аппроксимировать рядом первого порядка по η:

$$w_i(n+1) = w_i(n) + \eta y(n)[x_i(n) - y(n)w_i(n)]. \tag{8.40}$$

Слагаемое $y(n)x_i(n)$ в правой части (8.40) представляет собой обычную Хеббовскую модификацию синаптического веса w_i и, таким образом, участвует в процессе самоусиления, диктуемого первым принципом самоорганизации. Отрицательное слагаемое $(-y(n)w_i(n))$, согласно второму принципу, отвечает за стабилизацию. Оно преобразует входной сигнал $x_i(n)$ к форме, зависящей от соответствующего синаптического веса $w_i(n)$ и выходного сигнала $y(n)$:

$$x_i'(n) = x_i(n) - y(n)w_i(n). \tag{8.41}$$

Это выражение можно рассматривать как эффективный *входной сигнал* (effective input) i-го синапса. Теперь можно использовать определение (8.41) и переписать правило обучения (8.40) в следующем виде:

$$w_i(n+1) = w_i(n) + \eta y(n)x_i'(n). \tag{8.42}$$

Общая работа нейрона представляется как комбинация двух графов передачи сигнала, показанных на рис. 8.5. Граф на рис. 8.5, *а* отражает зависимость выхода $y(n)$ от синаптических весов $w_1(n), w_2(n), \ldots, w_m(n)$, согласно (8.36). Графы передачи сигнала на рис. 8.5, *б* иллюстрируют выражения (8.41) и (8.42). Передаточная функция z^{-1} в средней части графа представляет собой *оператор единичной задержки* (unit-delay operator). Выходной сигнал $y(n)$ на рис. 8.5, *а* выступает в роли передаточной функции на рис. 8.5, *б*. Граф на рис. 8.5, *б* ясно показывает следующие две формы внутренней обратной связи, действующей в нейроне.

- Положительная обратная связь для самоусиления и роста синаптических весов $w_i(n)$ в соответствии с внешним входным сигналом $x_i(n)$.
- Отрицательная обратная связь $(-y(n))$ для контроля роста связей, стабилизирующая синаптический вес $w_i(n)$.

Слагаемое-произведение $(-y(n)w_i(n))$ связано с *фактором забывания* (forgetting), который часто используется в правилах обучения, но с одним отличием: фактор забывания становится более явно выраженным с усилением выходного сигнала $y(n)$. Такой тип управления имеет определенное нейробиологическое объяснение [1016].

Матричная формулировка алгоритма

Для удобства выкладок введем следующие обозначения:

$$\mathbf{x}(n) = [x_1(n), x_2(n), \ldots, x_m(n)]^T \tag{8.43}$$

и

$$\mathbf{w}(n) = [w_1(n), w_2(n), \ldots, w_m(n)]^T. \tag{8.44}$$

Входной вектор $\mathbf{x}(n)$ и вектор синаптических весов $\mathbf{w}(n)$ обычно являются реализациями случайных векторов. Используя это векторное представление, выражение (8.36) можно переписать в форме скалярного произведения:

$$y(n) = \mathbf{x}^T(n)\mathbf{w}(n) = \mathbf{w}^T(n)\mathbf{x}(n). \tag{8.45}$$

Аналогично, выражение (8.40) можно переписать в следующем виде:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta y(n)[\mathbf{x}(n) - y(n)\mathbf{w}(n)]. \tag{8.46}$$

Подставляя (8.45) в (8.46), получим:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) - \mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n)\mathbf{w}(n)]. \tag{8.47}$$

Алгоритм обучения (8.47) представляет собой *нелинейное стохастическое разностное уравнение* (nonlinear stochastic difference equation), которое делает анализ сходимости этого алгоритма сложным с математической точки зрения. Для того чтобы обеспечить базис для анализа сходимости, немного отвлечемся от поставленной задачи и обратимся к общим методам анализа сходимости стохастических алгоритмов аппроксимации.

Теорема об асимптотической устойчивости

Алгоритм самоорганизующегося обучения, описываемый уравнением (8.47), является частным случаем общего алгоритма стохастической аппроксимации:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)h(\mathbf{w}(n), \mathbf{x}(n)), n = 0, 1, 2, \ldots . \tag{8.48}$$

Предполагается, что последовательность $\eta(n)$ состоит из положительных скаляров.

Функция коррекции (update function) $h(\cdot, \cdot)$ является детерминированной функцией с некоторыми условиями регулярности. Эта функция, вместе с последовательностью скаляров $\eta(\cdot)$, определяют полную структуру алгоритма.

Целью описываемой здесь процедуры является определение связи *детерминированного обычного дифференциального уравнения* (deterministic ordinary differential equation, или ODE) со стохастическим нелинейным разностным уравнением (8.48). После выявления такой связи свойства устойчивости этого дифференциального уравнения можно ассоциировать со свойствами сходимости алгоритма. Описываемая процедура является довольно общим приемом и имеет широкую область применения. Она была разработана независимо друг от друга в [665] и [607], использовавших совершенно разные подходы[2].

Для начала предположим, что алгоритм стохастической аппроксимации, описываемый уравнением (8.48), удовлетворяет следующим условиям.

1. Значения $\eta(n)$ — это убывающая последовательность положительных действительных чисел, такая, что:

   (а) $$\sum_{n=1}^{\infty} \eta(n) = \infty, \tag{8.49}$$

   (б) $$\sum_{n=1}^{\infty} \eta^p(n) < \infty, p > 1, \tag{8.50}$$

   (в) $$\eta(n) \to 0 \text{ при } n \to \infty. \tag{8.51}$$

2. Последовательность векторов параметров (синаптических весов) $\mathbf{w}(\cdot)$ является ограниченной с вероятностью 1.
3. Функция коррекции $h(\mathbf{w}, \mathbf{x})$ является непрерывно дифференцируемой по $\mathbf{w}$ и $\mathbf{x}$, а ее производные ограничены во времени.

---

[2] Подходы, примененные в [665] и [607], посвященных изучению динамики алгоритма стохастической аппроксимации, свели эту задачу к изучению диннамики дифференциальных уравнений. Однако эти два подхода отличаются друг от друга. В первой работе использовалась функция Ляпуиова, а во второй — процесс линейной интерполяции [270]. Последний подход был унаследован из [258] для изучения сходимости фильтра Хебба для извлечения максимального собственного значения. При этом полученные выводы совпали с результатами первого подхода.

4. Предел

$$\bar{h}(\mathbf{w}) = \lim_{n\to\infty} E[h(\mathbf{w}, \mathbf{X})] \tag{8.52}$$

существует для всех $\mathbf{w}$. Оператор статистического ожидания E действует на случайный вектор $\mathbf{X}$ с реализацией, обозначаемой символом $\mathbf{x}$.

5. Существует локально асимптотически устойчивое (в смысле Ляпунова) решение обычного дифференциального уравнения

$$\frac{d}{dt}\mathbf{w}(t) = \bar{h}(\mathbf{w}(t)), \tag{8.53}$$

где t — непрерывное время. Устойчивость в смысле Ляпунова рассматривается в главе 14.

6. Пусть $\mathbf{q}_1$ — решение уравнения (8.53) в области притяжения $\mathbf{B}(\mathbf{q})$. Понятие области притяжения или бассейна аттракции (basin of attraction) будет определено в главе 14. Тогда вектор параметров $\mathbf{w}(n)$ определяет компактное подмножество $\mathbf{A}$ в области притяжения $\mathbf{B}(\mathbf{q})$ бесконечно часто, с вероятностью 1.

Все шесть описанных выше условий являются обоснованными. В частности, условие 1(а) является необходимым условием способности алгоритма изменять оценку до требуемого значения, независимо от начального состояния; 1(б) определяет скорость сходимости $\eta(n)$ к нулю. Это ограничение является заметно более мягким, чем обычное

$$\sum_{n=1}^{\infty} \eta^2(n) < \infty.$$

Условие 4 является основным предположением, предоставляющим возможность ассоциировать дифференциальное уравнение с алгоритмом (8.48).

Теперь рассмотрим алгоритм стохастической аппроксимации, описываемый рекурсивным уравнением (8.48) при выполнении условий 1–6. Для данного класса стохастических алгоритмов аппроксимации можно сформулировать *теорему об асимптотической устойчивости* (asymptotic stability theorem) [607], [665]:

$$\lim_{n\to\infty} \mathbf{w}(n) = \mathbf{q}_1, \text{бесконечно часто с вероятностью 1.} \tag{8.54}$$

Однако необходимо подчеркнуть следующее. Несмотря на асимптотические свойства алгоритма (8.48), эта теорема ничего не говорит о количестве итераций n, необходимых для применения результатов этого анализа. Более того, в тех задачах, где

вектор параметров, зависящий от времени, должен отслеживаться с помощью алгоритма (8.48), выполнение условия 1(в) в принципе невозможно:

$$\eta(n) \to 0 \text{ при } n \to \infty.$$

Эту сложность можно обойти, назначая параметру η некоторое малое положительное значение, величина которого обычно зависит от области приложения. Это обычно происходит при практическом использовании алгоритма стохастической аппроксимации в нейронных сетях.

Анализ устойчивости фильтра для извлечения максимального собственного значения

С помощью анализа устойчивости ODE можно исследовать сходимость рекурсивного алгоритма (8.46), определяющего фильтр для извлечения *максимального собственного значения* (maximum eigenfilter).

Для того чтобы выполнялось условие 1 теоремы об асимптотической устойчивости, примем

$$\eta(n) = \frac{1}{n}.$$

Из выражения (8.47) видно, что функция коррекции $h(\mathbf{w}, \mathbf{x})$ определяется следующим образом:

$$\begin{aligned} h(\mathbf{w}, \mathbf{x}) &= \mathbf{x}(n)y(n) - y^2(n)\mathbf{w}(n) = \\ &= \mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) - [\mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n)]\mathbf{w}(n), \end{aligned} \tag{8.55}$$

что, очевидно, удовлетворяет условию 3 теоремы. Уравнение (8.55) является результатом использования реализации $\mathbf{x}$ случайного вектора $\mathbf{X}$ в функции коррекции $h(\mathbf{w}, \mathbf{X})$. Для выполнения условия 4 возьмем математическое ожидание $h(\mathbf{w}, \mathbf{X})$ по $\mathbf{X}$, таким образом, можно записать, что

$$\begin{aligned} \bar{h} &= \lim_{n\to\infty} E[\mathbf{X}(n)\mathbf{X}^T(n)\mathbf{w}(n) - (\mathbf{w}^T(n)\mathbf{X}(n)\mathbf{X}^T(n)\mathbf{w}(n))\mathbf{w}(n)] = \\ &= \mathbf{R}\mathbf{w}(\infty) - [\mathbf{w}^T(\infty)\mathbf{R}\mathbf{w}(\infty)]\mathbf{w}(\infty), \end{aligned} \tag{8.56}$$

где $\mathbf{R}$ — матрица корреляции стохастического процесса, представленного случайным вектором $\mathbf{X}(n)$; $\mathbf{w}(\infty)$ — предельное значение вектора синаптических весов.

Согласно условию 5, в свете уравнений (8.53) и (8.56) определим устойчивые точки нелинейного дифференциального уравнения

$$\frac{d}{dt}\mathbf{w}(t) = \bar{h}(\mathbf{w}(t)) = \mathbf{R}\mathbf{w}(t) - [\mathbf{w}^T(t)\mathbf{R}]\mathbf{w}(t). \tag{8.57}$$

Рассмотрим разложение вектора $\mathbf{w}(t)$ в терминах полного ортогонального множества собственных векторов матрицы корреляции $\mathbf{R}$ следующим образом:

$$\mathbf{w}(t) = \sum_{k=1}^{m} \theta_k(t)\mathbf{q}_k, \tag{8.58}$$

где $\mathbf{q}_k$ — k-й нормализованный собственный вектор матрицы $\mathbf{R}$; коэффициент $\theta_k(t)$ — проекция, зависящая от времени, вектора $\mathbf{w}(t)$ на $\mathbf{q}_k$. Подставим (8.58) в (8.57) и используем основные определения

$$\mathbf{R}\mathbf{q}_k = \lambda_k\mathbf{q}_k$$

и

$$\mathbf{q}_k^T\mathbf{R}\mathbf{q}_k = \lambda_k,$$

где λ_k — собственное значение, связанное с вектором $\mathbf{q}_k$. Окончательно получим:

$$\sum_{k=1}^{m} \frac{d\theta_k(t)}{dt}\mathbf{q}_k = \sum_{k=1}^{m} \lambda_k\theta_k(t)\mathbf{q}_k - \left[\sum_{l=1}^{m} \lambda_l\theta_l^2(t)\right] \sum_{k=1}^{m} \theta_k(t)\mathbf{q}_k. \tag{8.59}$$

Эквивалентно можно записать:

$$\frac{d\theta_k(t)}{dt} = \lambda_k\theta_k(t) - \theta_k(t)\sum_{l=1}^{m} \lambda_l\theta_l^2(t), \quad k = 1, 2, \ldots, m. \tag{8.60}$$

Таким образом, анализ сходимости алгоритма стохастической аппроксимации (8.48) сведен к анализу устойчивости системы обычных дифференциальных уравнений (8.60), содержащих *главные моды* (principal modes) $\theta_k(t)$.

Необходимо рассмотреть два случая, в зависимости от того, какое значение принимает индекс k. Первый случай соответствует диапазону $1 < k \leq m$, а второй — $k = 1$. Число m является размерностью векторов $\mathbf{x}(n)$ и $\mathbf{w}(n)$. Оба этих случая детально рассматриваются ниже.

Случай 1. $1 < k \leq m$.

Для рассмотрения этого случая определим:

$$\alpha_k(t) = \frac{\theta_k(t)}{\theta_1(t)}, \quad 1 < k \leq m. \tag{8.61}$$

Исходя из этого соотношения, предполагается, что $\theta_1(t) \neq 0$ с вероятностью 1, принимая во внимание, что *начальные значения* $\mathbf{w}(0)$ *выбираются случайным образом*. Затем, дифференцируя по t обе части уравнения (8.61), получим:

$$\begin{aligned} \frac{d\alpha_k(t)}{dt} &= \frac{1}{\theta_1(t)}\frac{d\theta_k(t)}{dt} - \frac{\theta_k(t)}{\theta_1^2(t)}\frac{d\theta_1(t)}{dt} = \\ &= \frac{1}{\theta_1(t)}\frac{d\theta_k(t)}{dt} - \frac{\alpha_k(t)}{\theta_1(t)}\frac{d\theta_1(t)}{dt}, \quad 1 < k \leq m. \end{aligned} \tag{8.62}$$

Далее, подставляя (8.60) в (8.62), применяя определение (8.61) и затем упрощая результат, получим:

$$\frac{d\alpha_k(t)}{dt} = -(\lambda_1 - \lambda_k)\alpha_k(t), \quad 1 < k \leq m. \tag{8.63}$$

Предполагая, что собственные значения матрицы корреляции $\mathbf{R}$ являются различными и упорядочены по убыванию, получим:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_k > \ldots > \lambda_m > 0. \tag{8.64}$$

Отсюда следует, что разность между собственными значениями $\lambda_1 - \lambda_k$, представляющая аналог константы времени в (8.63), является положительной. Поэтому для случая 1 выполняется соотношение

$$\alpha_k(t) \to 0 \text{ при } t \to \infty \text{ для } 1 < k \leq m. \tag{8.65}$$

Случай 2. $k = 1$.

Исходя из выражения (8.60), этот второй случай описывается следующим дифференциальным уравнением:

$$\begin{aligned} \frac{d\theta_1(t)}{dt} &= \lambda_1\theta_1(t) - \theta_1(t)\sum_{l=1}^{m}\lambda_l\theta_l^2(t) = \lambda_1\theta_1(t) - \lambda_1\theta_1^3(t) - \theta_1(t)\sum_{l=2}^{m}\lambda_l\theta_l^2(t) = \\ &= \lambda_1\theta_1(t) - \lambda_1\theta_1^3(t) - \theta_1^3(t)\sum_{l=2}^{m}\lambda_l\alpha_l^2(t). \end{aligned} \tag{8.66}$$

Однако из случая 1 известно, что $\alpha_l \to 0$ для $l \neq 1$ при $t \to \infty$. Следовательно, последнее слагаемое в (8.66) достигает нуля при достижении времени $t \to \infty$. Игнорируя это слагаемое, выражение (8.66) можно упростить:

$$\frac{d\theta_1(t)}{dt} = \lambda_1 \theta_1(t)[1 - \theta_1^2(t)] \quad \text{для } t \to \infty. \tag{8.67}$$

Здесь следует подчеркнуть, что равенство (8.67) выполняется только в асимптотическом смысле.

Равенство (8.67) представляет собой *автономную систему* (autonomous system), т.е. систему, которая явно не зависит от времени. Устойчивость такой системы лучше всего обеспечивать с помощью положительно определенной функции, называемой *функцией Ляпунова*. Ее детальное рассмотрение будет предложено в главе 14. Пусть $\mathbf{s}$ — вектор состояния автономной системы, а $V(t)$ — функция Ляпунова этой системы. Равновесное состояние $\bar{\mathbf{s}}$ такой системы является асимптотически устойчивым, если

$$\frac{d}{dt}V(t) < 0 \text{ для } \mathbf{s} \in \mathbf{U} - \bar{\mathbf{s}},$$

где $\mathbf{U}$ — малая окрестность точки $\bar{\mathbf{s}}$.

Для нашей задачи докажем, что дифференциальное уравнение (8.67) имеет функцию Ляпунова следующего вида:

$$V(t) = [\theta_1^2(t) - 1]^2. \tag{8.68}$$

Для того чтобы проверить это утверждение, следует показать, что $V(t)$ удовлетворяет двум условиям:

1. $$\frac{dV(t)}{dt} < 0 \text{ для всех } t. \tag{8.69}$$

2. $$V(t) \text{ имеет минимум.} \tag{8.70}$$

Дифференцируя (8.68) по времени, получим:

$$\begin{aligned} \frac{dV(t)}{dt} &= 4\theta_1(t)[\theta_1(t) - 1]\frac{d\theta_1}{dt} = \\ &= -4\lambda_1\theta_1^2(t)[\theta_1^2(t) - 1]^2 \quad \text{для } t \to \infty, \end{aligned} \tag{8.71}$$

где во второй строке используется равенство (8.67). Так как собственное значение λ_1 является положительным, то с учетом (8.71) видно, что условие (8.69) истинно при достижении $t \to \infty$. Более того, из выражения (8.71) можно заметить, что $V(t)$ имеет

минимум (т.е. $dV(t)/dt = 0$) при $\theta_1(t) = \pm 1$), и, таким образом, условие (8.70) также удовлетворяется. Исходя из этого, можно завершить рассмотрение случая 2 следующим утверждением:

$$\theta_1(t) \to \pm 1 \text{ при } t \to \infty. \tag{8.72}$$

В свете результатов, представленных соотношением (8.72), и определения (8.71) можно еще раз записать утверждение, полученное для случая 1 и представленное выражением (8.65), в итоговом виде:

$$\theta_k(t) \to 0 \text{ при } t \to \infty \text{ для } 1 < k \leq m. \tag{8.73}$$

Общее заключение, которое следует из анализа случаев 1 и 2, можно выразить следующим образом.

- Единственной модой алгоритма стохастической аппроксимации, описанной выражением (8.47), которая сходится, является $\theta_1(t)$. Все остальные моды этого алгоритма будут сводиться к нулю.
- Мода $\theta_1(t)$ сходится к ± 1.

Таким образом, выполняется условие 5 теоремы об асимптотической устойчивости. В частности, в свете разложения, описываемого выражением (8.58), можно формально утверждать, что

$$\mathbf{w}(t) \to \mathbf{q}_1 t \to \infty,$$

где $\mathbf{q}_1$ является нормализованным собственным вектором, ассоциированным с наибольшим собственным значением матрицы корреляции $\mathbf{R}$.

Далее необходимо показать, что в соответствии с условием 6 теоремы об асимптотической устойчивости существует такое подмножество $\mathbf{A}$ множества всех векторов, что

$$\lim_{n \to \infty} \mathbf{w}(n) = \mathbf{q}_1 \text{ бесконечно часто с вероятностью 1.}$$

Для этого прежде всего нужно обеспечить выполнение условия 2, чего можно достичь с помощью *жестких ограничений*, налагаемых на $\mathbf{w}(n)$. Норма этого вектора должна оставаться ниже некоторого определенного порога a. Норму $\mathbf{w}(n)$ можно определить следующим образом:

$$\|\mathbf{w}(n)\| = \max_j |w_j(n)| \leq a. \tag{8.74}$$

Пусть $\mathbf{A}$ — компактное подмножество пространства $\Re^m$, определенное множеством векторов с нормой, не превышающей порога a. Несложно показать следующее [932].

Если $||\mathbf{w}(n)|| \leq a$ *и константа* a *достаточно велика, то* $||\mathbf{w}(n+1)|| < ||\mathbf{w}(n)||$ *с вероятностью 1.*

Таким образом, по мере увеличения количества итераций n, $\mathbf{w}(n)$ может случайно оказаться внутри $\mathbf{A}$ (бесконечно часто) с вероятностью 1. Так как область притяжения $\mathbf{B}(\mathbf{q}_1)$ включает все векторы с ограниченной нормой, то $\mathbf{A} \in \mathbf{B}(\mathbf{q}_1)$. Другими словами, условие 6 выполняется.

Итак, мы обеспечили выполнение всех шести условий теоремы об асимптотической сходимости и показали, что (при вышеупомянутых условиях) алгоритм стохастической аппроксимации (8.47) гарантирует сходимость $\mathbf{w}(n)$ с вероятностью 1 к собственному вектору $\mathbf{q}_1$, связанному с наибольшим собственным значением λ_1 матрицы корреляции $\mathbf{R}$. Это не единственная *стационарная* (fixed) точка алгоритма, но только она является асимптотически устойчивой.

Общие свойства фильтра Хебба для извлечения максимального собственного значения

Представленный вниманию читателя анализ сходимости показывает, что единственный самоорганизующийся нейрон, работающий под управлением правила обучения (8.39), или, что эквивалентно, (8.46), адаптивно извлекает первый главный компонент из стационарного входного сигнала. Первый главный компонент соответствует наибольшему собственному числу матрицы корреляции случайного вектора $\mathbf{X}(n)$. В действительности значение λ_1 связано с дисперсией выходного сигнала модели $y(n)$.

Пусть $\sigma^2(n)$ — дисперсия случайной переменной $Y(n)$, реализация которой обозначается как $y(n)$, т.е.

$$\sigma^2(n) = E[Y^2(n)], \tag{8.75}$$

где $Y(n)$ имеет нулевое среднее значение для входного сигнала с нулевым средним. Устремляя в (8.46) n к бесконечности и используя тот факт, что при этом условии $\mathbf{w}(n)$ достигает значения $\mathbf{q}_1$, получим:

$$\mathbf{x}(n) = y(n)\mathbf{q}_1 \text{ при } n \to \infty.$$

Используя это соотношение, можно показать, что дисперсия $\sigma^2(n)$ достигает значения λ_1 при количестве итераций n, стремящемся к бесконечности (см. задачу 8.2).

В результате линейный Хеббоподобный нейрон, работа которого описывается выражением (8.46), сходится с вероятностью 1 к стационарной точке, которая характеризуется следующим [798].

Дисперсия выхода модели достигает наибольшего собственного значения матрицы корреляции $\mathbf{R}$:

$$\lim_{n\to\infty} \sigma^2(n) = \lambda_1. \tag{8.76}$$

Вектор синаптических весов этой модели достигает соответствующего собственного вектора:

$$\lim_{n\to\infty} \mathbf{w}(n) = \mathbf{q}_1 \tag{8.77}$$

при

$$\lim_{n\to\infty} \|\mathbf{w}(n)\| = 1. \tag{8.78}$$

Эти результаты предполагают, что матрица корреляции $\mathbf{R}$ является положительно определенной, а наибольшее собственное число λ_1 имеет кратность 1. Они также имеют место для неотрицательно определенной матрицы $\mathbf{R}$, если $\lambda_1>0$ и имеет кратность 1.

Пример 8.2

Согласованный фильтр (matched filter)

Рассмотрим случайный вектор $\mathbf{X}(n)$, составленный следующим образом:

$$\mathbf{X}(n) = \mathbf{s} + \mathbf{V}(n),$$

где $\mathbf{s}$ — фиксированный единичный вектор, представляющий *компонент полезного сигнала* (signal component); $\mathbf{V}(n)$ — компонент *белого шума* со средним значением 0. Матрица корреляции такого входного вектора имеет следующий вид:

$$\mathbf{R} = E[\mathbf{X}(n)\mathbf{X}^T(n)] = \mathbf{s}\mathbf{s}^T + \sigma^2\mathbf{I},$$

где σ^2 — дисперсия элементов вектора шума $\mathbf{V}(n)$; $\mathbf{I}$ — единичная матрица. Наибольшее собственное значение такой матрицы будет следующим:

$$\lambda_1 = 1 + \sigma^2.$$

Соответствующий собственный вектор $\mathbf{q}_1$ имеет вид

$$\mathbf{q}_1 = \mathbf{s}.$$

Не составляет труда показать, что это решение удовлетворяет уравнению для собственных значений:

$$\mathbf{R}\mathbf{q}_1 = \lambda\mathbf{q}_1.$$

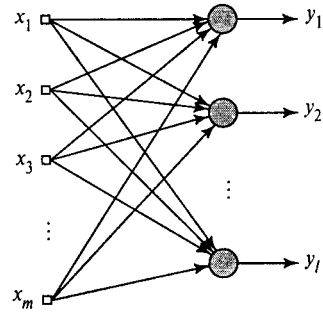


Рис. 8.6. Сеть прямого распространения с одним слоем вычислительных элементов

Для описанной в данном примере ситуации самоорганизующийся линейный нейрон (при его сходимости к устойчивому состоянию) выступает в качестве *согласованного фильтра* (matched filter) в том смысле, что его импульсный отклик (представленный синаптическими весами) соответствует компоненту полезного сигнала **s** входного вектора $\mathbf{X}(n)$. ■

8.5. Анализ главных компонентов на основе фильтра Хебба

Описанный в предыдущем разделе фильтр Хебба извлекает первый главный компонент из входного сигнала. Линейная модель с одним нейроном может быть расширена до сети прямого распространения с одним слоем линейных нейронов с целью анализа главных компонентов для входного сигнала произвольной размерности [932].

Для большей конкретизации рассмотрим сеть прямого распространения, показанную на рис. 8.6. В ней сделаны следующие допущения относительно структуры.

1. Все нейроны выходного слоя сети являются *линейными*.
2. Сеть имеет m входов и l выходов. Более того, количество выходов меньше количества входов (т.е. $l < m$).

Обучению подлежит только множество синаптических весов $\{w_{ji}\}$, соединяющих узлы i входного слоя с вычислительными узлами j выходного слоя, где $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, l$.

Выходной сигнал $y_j(n)$ нейрона j в момент времени n, являющийся откликом на множество входных воздействий $\{x_i(n) | i = 1, 2, \ldots, m\}$, определяется по следующей формуле (рис. 8.7, *а*):

$$y_j(n) = \sum_{i=1}^{m} w_{ji}(n) x_i(n), \quad j = 1, 2, \ldots, l. \tag{8.79}$$

Синаптический вес $w_{ji}(n)$ настраивается в соответствии с обобщенной формой правила обучения Хебба [932]:

$$\Delta w_{ji}(n) = \eta \left[y_j(n)x_i(n) - y_j(n) \sum_{k=1}^{j} w_{ki}(n)y_k(n) \right], \begin{matrix} i = 1, 2, \ldots, m; \\ j = 1, 2, \ldots, l, \end{matrix} \tag{8.80}$$

где $\Delta w_{ji}(n)$ — коррекция, применяемая к синаптическому весу $w_{ji}(n)$ в момент времени n; η — параметр скорости обучения. *Обобщенный алгоритм обучения Хебба* (generalized Hebbian algorithm — GHA) (8.80) для слоя из l нейронов включает в себя алгоритм (8.39) для одного нейрона в качестве частного случая, т.е. для $l = 1$.

Для того чтобы заглянуть вглубь обобщенного алгоритма обучения Хебба, перепишем уравнение (8.80) в следующем виде:

$$\Delta w_{ji}(n) = \eta\, y_j(n) \left[x_i'(n) - w_{ji}(n)y_j(n) \right], \begin{matrix} i = 1, 2, \ldots, m; \\ j = 1, 2, \ldots, l, \end{matrix} \tag{8.81}$$

где $x_i'(n)$ — модифицированная версия i-го элемента входного вектора $\mathbf{x}(n)$, являющаяся функцией индекса j, т.е.

$$x_i'(n) = x_i(n) - \sum_{k=1}^{j-1} w_{ki}(n)y_k(n). \tag{8.82}$$

Для конкретного нейрона j алгоритм, описанный выражением (8.81), имеет ту же математическую форму, что и (8.39), за исключением того факта, что в (8.82) входной сигнал $x_i(n)$ заменен его модифицированным значением $x_i'(n)$. Теперь можно сделать следующий шаг и переписать выражение (8.81) в форме, соответствующей постулату обучения Хебба:

$$\Delta w_{ji}(n) = \eta y_j(n)x_i''(n), \tag{8.83}$$

где

$$x_i''(n) = x_i' - w_{ji}(n)y_j(n). \tag{8.84}$$

Таким образом, принимая во внимание

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{8.85}$$

и

$$w_{ji}(n) = z^{-1}[w_{ji}(n+1)], \tag{8.86}$$

где z^{-1} — оператор единичной задержки, можно построить граф передачи сигнала, показанный на рис. 8.7, *б*, для обобщенного алгоритма Хебба. Из этого графа видно,

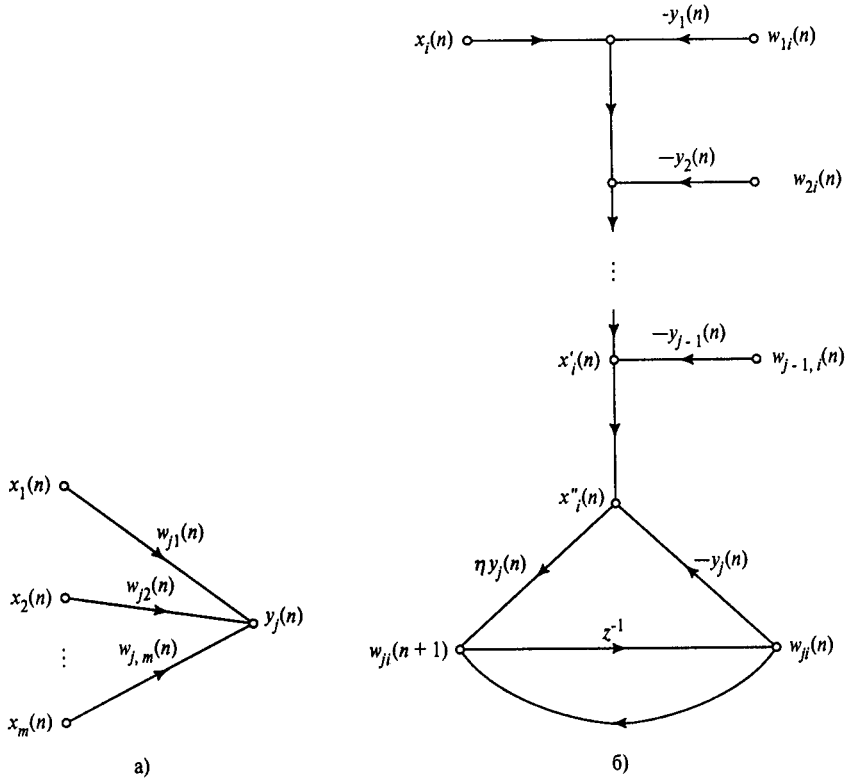


Рис. 8.7. Представление обобщенного алгоритма Хебба в виде графа передачи сигнала: граф уравнения (8.79) (а); граф выражений (8.80), (8.81) (б)

что сам алгоритм (согласно его формулировке в (8.85)) базируется на *локальной* форме реализации. Учтем также, что выход $y_j(n)$, отвечающий за обратную связь на графе передачи сигнала (см. рис. 8.7, *б*), определяется по формуле (8.79). Представление последнего уравнения в виде графа передачи сигнала показано на рис. 8.7, *а*.

Для эвристического понимания того, как обобщенный алгоритм Хебба работает на самом деле, в первую очередь запишем версию алгоритма (8.81) в матричном представлении:

$$\Delta \mathbf{w}_j(n) = \eta y_j(n)\mathbf{x}'(n) - \eta y_j^2(n)\mathbf{w}_j(n), \quad j = 1, 2, \ldots, l, \tag{8.87}$$

где

$$\mathbf{x}'(n) = \mathbf{x}(n) - \sum_{k=1}^{j-1} \mathbf{w}_k(n) y_k(n). \tag{8.88}$$

Вектор $\mathbf{x}'(n)$ представляет собой модифицированную форму входного вектора. Основываясь на представлении (8.87), можно сделать следующие наблюдения [932].

Для первого нейрона сети прямого распространения, показанной на рис. 8.6,

$$j = 1 : \mathbf{x}'(n) = \mathbf{x}(n).$$

Для этого случая обобщенный алгоритм Хебба сводится к виду (8.46), записанному для одиночного нейрона. Из материала, представленного в разделе 8.5, известно, что этот нейрон извлекает первый основной компонент входного вектора $\mathbf{x}(n)$.

1. Для второго нейрона сети на рис. 8.6 можно записать:

$$j = 2 : \mathbf{x}'(n) = \mathbf{x}(n) - \mathbf{w}_1(n)y_1(n).$$

   Учитывая, что первый нейрон уже извлек первый главный компонент, второй нейрон видит входной вектор $\mathbf{x}'(n)$, из которого уже удален первый собственный вектор матрицы корреляции $\mathbf{R}$. Таким образом, второй нейрон извлекает первый главный компонент $\mathbf{x}'(n)$, что эквивалентно второму главному компоненту исходного входного вектора $\mathbf{x}(n)$.
2. Для третьего нейрона можно записать:

$$j = 3 : \mathbf{x}'(n) = \mathbf{x}(n) - \mathbf{w}_1(n)y_1(n) - \mathbf{w}_2(n)y_2(n).$$

   Предположим, что первые два нейрона уже сошлись к первому и второму главным компонентам (см. пп. 1 и 2). Значит, третий нейрон видит входной вектор $\mathbf{x}'(n)$, из которого удалены первый и второй собственные векторы. Таким образом, он извлекает первый главный компонент вектора $\mathbf{x}'(n)$, что эквивалентно третьему главному компоненту исходного входного вектора $\mathbf{x}(n)$.
3. Продолжая эту процедуру для оставшихся нейронов сети прямого распространения (см. рис. 8.6), получим, что каждый из выходов сети, обученный с помощью обобщенного алгоритма Хебба (8.81), представляет собой отклик на конкретный собственный вектор матрицы корреляции входного вектора, причем отдельные выходы упорядочены по убыванию ее собственных значений.

Этот метод вычисления собственных векторов аналогичен методу, получившему название *процесса исчерпания* [599]. Он использует процедуру, аналогичную ортогонализации Грама–Шмидта [1022].

Представленное здесь описание "от нейрона к следующему нейрону" было приведено для упрощения изложения. На практике все нейроны в обобщенном алгоритме Хебба совместно работают на обеспечение сходимости.

Исследование сходимости

Пусть $\mathbf{W}(n) = \{w_{ji}(n)\}$ — матрица весов размерности $m \times l$ сети прямого распространения (рис. 8.6):

$$\mathbf{W}(n) = [\mathbf{w}_1(n), \mathbf{w}_2(n), \ldots, \mathbf{w}_l(n)]^T. \tag{8.89}$$

Пусть параметр скорости обучения обобщенного алгоритма Хебба (8.81) имеет форму, зависящую от времени $\eta(n)$, такую, что в пределе

$$\lim_{n\to\infty} \eta(n) = 0 \text{ и } \sum_{n=0}^{\infty} \eta(n) = \infty. \tag{8.90}$$

Тогда этот алгоритм можно переписать в матричном виде:

$$\Delta\mathbf{W}(n) = \eta(n)\{\mathbf{y}(n)\mathbf{x}^T(n) - \mathrm{LT}[\mathbf{y}(n)\mathbf{y}^T(n)]\mathbf{W}(n)\}, \tag{8.91}$$

где оператор LT[·] устанавливает все элементы, расположенные выше диагонали матрицы аргументов, в нуль. Таким образом, полученная матрица становится *нижней треугольной* (lower triangular). При этих условиях и допущениях, изложенных в разделе 8.4, сходимость алгоритма GHA доказывается с помощью процедуры, аналогичной представленной в предыдущем разделе для фильтра по извлечению максимального собственного значения. В связи с этим можно сформулировать следующую теорему [932].

Если элементы матрицы синаптических весов $\mathbf{W}(n)$ на шаге $n = 0$ принимают случайные значения, то с вероятностью 1 обобщенный алгоритм Хебба (8.91) будет сходиться к фиксированной точке, а $\mathbf{W}^T(n)$ достигнет матрицы, столбцы которой являются первыми l собственными векторами матрицы корреляции $\mathbf{R}$ размерности $m \times m$ входных векторов размерности $m \times 1$, упорядоченных по убыванию собственных значений.

Практическое значение этой теоремы состоит в том, что она гарантирует нахождение обобщенным алгоритмом Хебба первых l собственных векторов матрицы корреляции $\mathbf{R}$, в предположении, что соответствующие собственные значения отличны друг от друга. При этом важен и тот факт, что в данном случае не требуется вычислять саму матрицу корреляции $\mathbf{R}$: ее первые l собственных векторов вычисляются непосредственно на основании входных данных. Полученная экономия вычислительных ресурсов может быть особенно большой, если размерность входного пространства m достаточно велика, а требуемое количество собственных векторов, связанных с l наибольшими собственными значениями матрицы корреляции $\mathbf{R}$, является лишь небольшой частью размерности m.

Данная теорема о сходимости сформулирована в терминах зависящего от времени параметра скорости обучения $\eta(n)$. На практике этот параметр обычно принимает значение некоторой малой константы η. В этом случае сходимость гарантируется в смысле среднеквадратической ошибки синаптических весов порядка η.

В [181] исследовались свойства сходимости алгоритма GHA (8.91). Проведенный в работе анализ показал, что увеличение параметра η ведет к более быстрой сходимости и увеличению асимптотической среднеквадратической ошибки (что интуитивно предполагалось). Среди прочего в этой работе точно показана обратная зависимость между точностью вычислений и скоростью обучения.

Оптимальность обобщенного алгоритма Хебба

Предположим, что в пределе можно записать:

$$\Delta \mathbf{w}_j(n) \to \mathbf{0} \text{ и } \mathbf{w}_j(n) \to \mathbf{q}_j \text{ при } n \to \infty, \text{ где } j = 1, 2, \ldots, l \tag{8.92}$$

и

$$||\mathbf{w}_j(n)|| = 1 \text{ для всех } j. \tag{8.93}$$

Тогда предельные значения $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_l$ векторов синаптических весов нейронов сети прямого распространения (см. рис. 8.5) представляют собой *нормализованные собственные векторы* (normalized eigenvector), ассоциированные с l доминирующими собственными значениями матрицы корреляции $\mathbf{R}$, упорядоченными по убыванию собственных значений. Таким образом, для точки равновесия можно записать следующее:

$$\mathbf{q}_j^T \mathbf{R} \mathbf{q}_k = \begin{cases} \lambda_j, & k = j, \\ 0, & k \neq j, \end{cases} \tag{8.94}$$

где $\lambda_1 > \lambda_2 > \ldots > \lambda_l$.

Для выхода нейрона j получим предельное значение:

$$\lim_{n \to \infty} y_j(n) = \mathbf{x}^T(n) \mathbf{q}_j = \mathbf{q}_j^T \mathbf{x}(n). \tag{8.95}$$

Пусть $Y_j(n)$ — случайная переменная с реализацией $y_j(n)$. *Взаимная корреляция* (cross-correlation) между случайными переменными $Y_j(n)$ и $Y_k(n)$ в равновесном состоянии записывается в виде

$$\lim_{n \to \infty} E[Y_j(n) Y_k(n)] = E[\mathbf{q}_j^T \mathbf{X}(n) \mathbf{q}_k] = \mathbf{q}_j^T \mathbf{R} \mathbf{q}_k = \begin{cases} \lambda_j, & k = j, \\ 0, & k \neq j. \end{cases} \tag{8.96}$$

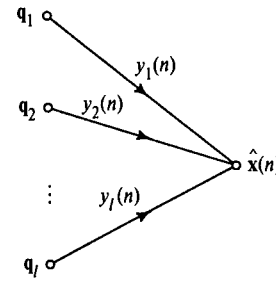


Рис. 8.8. Представление в виде графа передачи сигнала процесса восстановления вектора $\hat{\mathbf{x}}$

Следовательно, можно утверждать, что в точке равновесия обобщенный алгоритм Хебба (8.91) выступает в роли средства *собственных значений* (eigen-analyzer) входных данных.

Пусть $\hat{\mathbf{x}}(n)$ — частное значение входного вектора $\mathbf{x}(n)$, для которого предельные условия (8.92) удовлетворяются при $j = l - 1$. Тогда из матричной формы (8.80) можно получить, что в пределе

$$\hat{\mathbf{x}}(n) = \sum_{k=1}^{l} y_k(n)\mathbf{q}_k. \tag{8.97}$$

Это значит, что для заданных двух множеств величин — предельных значений $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_l$ векторов синаптических весов нейронов сети прямого распространения (см. рис. 8.5) и соответствующих выходных сигналов $y_1, y_2, \ldots, y_l$ — можно построить *линейную оценку по методу наименьших квадратов* (linear least-squares estimate) значения $\hat{\mathbf{x}}(n)$ входного вектора $\mathbf{x}(n)$. В результате формулу (8.97) можно рассматривать как одну из форм *восстановления данных* (data reconstruction) (рис. 8.8). Обратите внимание, что в свете дискуссии, представленной в разделе 8.3, этот метод восстановления данных имеет вектор ошибки аппроксимации, ортогональный оценке $\hat{\mathbf{x}}(n)$.

Алгоритм GHA в сжатом виде

Вычисления, выполняемые обобщенным алгоритмом Хебба (GHA), являются простыми, и их можно описать следующей последовательностью действий.

1. В момент времени $n = 1$ инициализируем синаптические веса w_{ji} сети случайными малыми значениями. Назначаем параметру скорости обучения η некоторое малое положительное значение.
2. Для $n = 1, j = 1, 2, \ldots, l$ и $i = 1, 2, \ldots, m$ вычислим:

$$y_j(n) = \sum_{i=1}^{m} w_{ji}(n)x_i(n),$$

$$\Delta w_{ji}(n) = \eta \left[y_j(n)x_i(n) - y_j(n) \sum_{k=1}^{j} w_{ki}(n) y_k(n) \right],$$

где $x_i(n)$ — i-й компонент входного вектора $\mathbf{x}(n)$ размерности $m \times 1$; l — требуемое число главных компонентов.

3. Увеличиваем значение n на единицу, переходим к шагу 2 и продолжаем до тех пор, пока синаптические веса w_{ji} не достигнут своих *установившихся* (steady-state) значений. Для больших n синаптические веса w_{ji} нейрона j сходятся к i-му компоненту собственного вектора, связанного с j-м собственным значением матрицы корреляции входного вектора $\mathbf{x}(n)$.

8.6. Компьютерное моделирование: кодирование изображений

В завершение рассмотрения обобщенного алгоритма обучения Хебба проверим его работу при решении задачи *кодирования изображений* (image coding). На рис. 8.9, *а* показана семейная фотография, использованная для обучения. Обратите внимание на *границы* (edge) фрагментов изображения. Это изображение формата 256 × 256 с 256 градациями серого цвета. Изображение было закодировано с помощью линейной сети прямого распространения, состоящей из одного слоя, содержащего 8 нейронов, каждый из которых имеет по 64 входа. Для обучения сети использовались непересекающиеся блоки размером 8 × 8. Эксперимент проводился для 2000 образцов сканирования этого изображения и маленького значения параметра скорости обучения, $\eta = 10^{-4}$.

На рис. 8.9, *б* показаны маски размера 8 × 8, представляющие синаптические веса сети. Каждая из восьми масок отображает множество синаптических весов, связанных с конкретным нейроном сети. В частности, возбуждающие синапсы (положительные веса) показаны белым цветом, тормозящие (отрицательные значения) — черным, а серым цветом показаны нулевые веса. В наших обозначениях маски представляют собой столбцы матрицы $\mathbf{W}^T$, состоящей из 64 × 8 синаптических весов, полученной после сходимости обобщенного алгоритма обучения Хебба.

Для кодирования изображения использовалась следующая процедура.

- Каждый из блоков изображения размером 8 × 8 пикселей был умножен на каждую из 8 масок, показанных на рис. 8.9, *б*. Таким образом, генерируются 8 коэффициентов кодирования изображения. На рис. 8.9, *в* показано реконструированное изображение, основанное на 8 доминирующих компонентах без дискретизации.

- Каждый из коэффициентов был равномерно дискретизирован на множестве битов, приблизительно пропорциональном логарифму дисперсии этого коэффициента на изображении. Таким образом, первым трем маскам было выделено по 6 бит, следующим двум — по 4, следующим двум — по 3, а оставшейся маске — 2 бита. На основе этого представления для кодирования каждого из блоков размерности 8×8 было выделено 34 бита, что соответствует уровню сжатия 0,53 бита на пиксель.

Для восстановления изображения на основе дискретных коэффициентов все маски были взвешены своими коэффициентами дискретизации, после чего они использовались для восстановления каждого из блоков рисунка. Реконструированная семейная фотография с уровнем сжатия 15:1 показана на рис. 8.9, *г*.

В качестве второго примера для иллюстрации обобщенного алгоритма обучения Хебба рассмотрим изображение морского пейзажа (рис. 8.10, *а*). На этом рисунке внимание акцентируется на *текстурной* информации. На рис. 8.10, *б* показаны маски размером 8×8 синаптических весов сети. Выполняемая процедура аналогична описанной выше. Обратите внимание на различия в масках на рис. 8.10, *б* и рис. 8.9, *б*. На рис. 8.10, *в* показано реконструированное изображение морского пейзажа, основанное на 8 доминирующих главных компонентах без дискретизации. Для изучения эффекта от дискретизации выходы первых двух масок были дискретизированы с использованием 5 бит для каждой, третья — с использованием 3 бит, а оставшиеся маски — с использованием 2 бит для каждой. Таким образом, для кодирования каждого из блоков 8×8 потребовалось 23 бита. В результате при кодировании в среднем использовалось 0,36 бита на пиксель. На рис. 8.10, *г* показано восстановленное изображение морского пейзажа, в котором используются маски, дискретизированные описанным выше способом. Общий уровень сжатия составил 22:1.

Для проверки общей эффективности "обобщения" обобщенного алгоритма Хебба использовались маски, показанные на рис. 8.9, *б*, для декомпозиции изображения морского пейзажа (см. рис. 8.10, *а*). При этом применялась та же процедура, которая использовалась для восстановления рис. 8.10, *г*. Результат этого восстановленного изображения показан на рис. 8.10, *д*, с уровнем сжатия 22:1, т.е. с тем же уровнем, который был получен для рис. 8.10, *г*. Несмотря на то что рис. 8.10, *г* и *д* имеют поразительное сходство, мы видим, что на первом из них более "правдиво" представлена текстурная информация. В результате этот рисунок выглядит менее "блочным", чем рис. 8.10, *д*. Это различие объясняется разными весами сети. Первые четыре веса, полученные после обучения нейронной сети на изображениях с семейной фотографией и морским пейзажем, являются довольно сходными. Однако последние четыре веса, кодирующие для первого рисунка информацию о контурах, в случае с морским пейзажем содержат информацию о текстуре. Таким образом, для восстановления текстурной информации морского пейзажа применялись "контурные" веса, что и привело к блочному представлению.

Рис. 8.9. Семейная фотография, использованная в эксперименте по кодированию изображений (а); маски размером 8×8, представляющие собой синаптические веса, обучаемые по алгоритму GHA (б); семейная фотография, восстановленная с помощью 8 доминирующих главных компонентов без дискретизации (в); реконструированная фотография с уровнем сжатия 15:1 при использовании дискретизации (г)

8.7. Адаптивный анализ главных компонентов с использованием латерального торможения

Обобщенный алгоритм Хебба, описанный в предыдущем разделе, основан исключительно на использовании прямых связей для анализа главных компонентов. В этом разделе мы опишем еще один алгоритм, называемый алгоритмом *адаптивного извлечения главных компонентов* (adaptive principal components extraction — APEX)

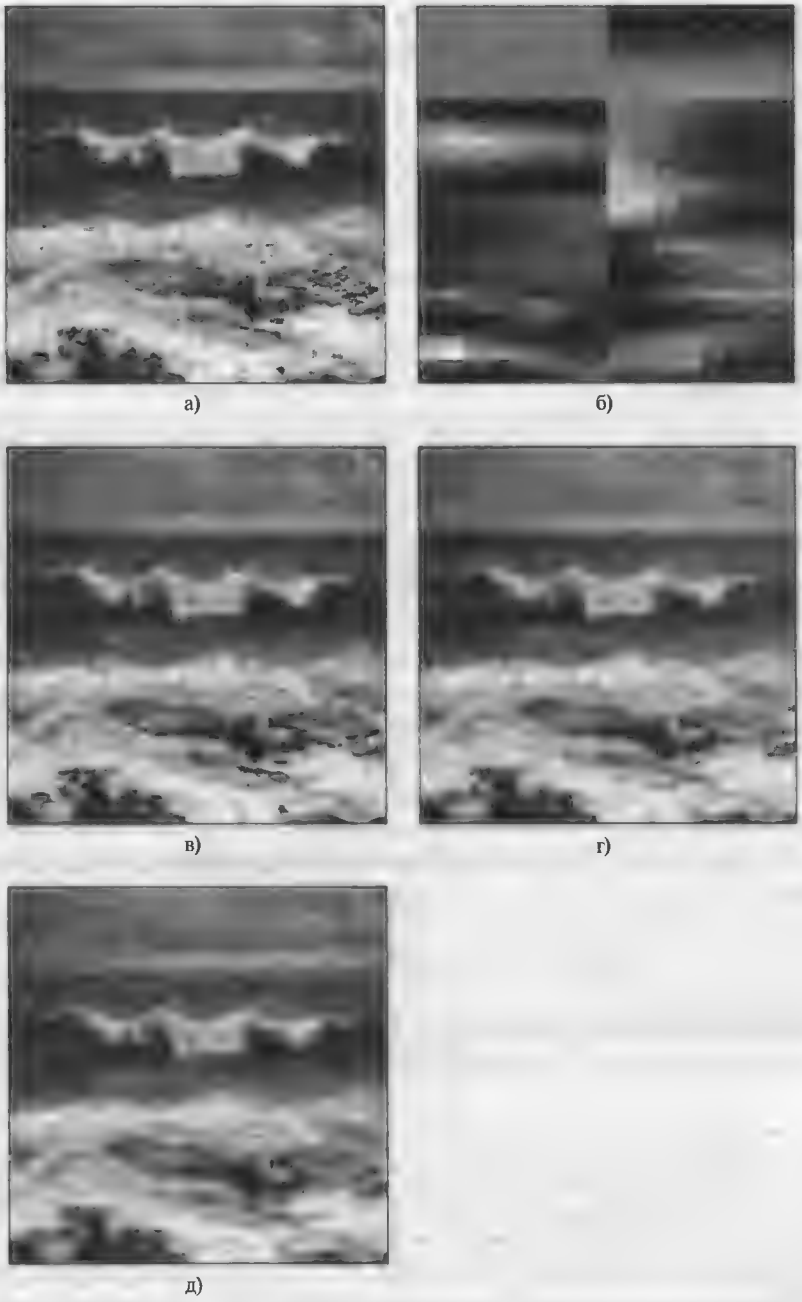


Рис. 8.10. Изображение морского пейзажа (а); маски 8×8, представляющие синаптические веса, обучаемые по алгоритму HGA и применяемые к изображению (б); восстановленное изображение морского пейзажа с использованием 8 доминирующих главных компонентов (в); восстановленное изображение с уровнем сжатия 22:1 при использовании масок из пункта (б) без дискретизации (г); восстановленное изображение морского пейзажа с использованием масок, показанных на рис. 8.9, *б*, с дискретизацией для достижения уровня сжатия 22:1, такого же, как в частях (г), (д)

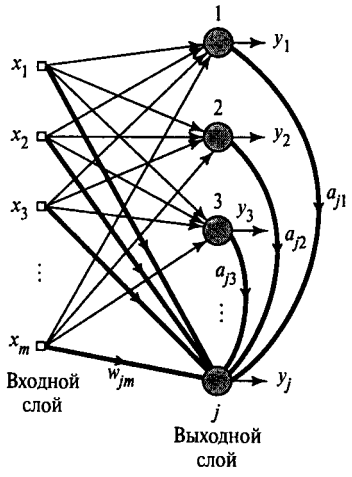


Рис. 8.11. Сеть с прямыми и латеральными связями, используемая для вывода алгоритма APEX

[258], [606]. Алгоритм APEX использует как прямые, так и обратные связи[3]. По своей природе этот алгоритм является итеративным: j-й основной компонент вычисляется на основе заданных (j-1) предыдущих.

На рис. 8.11 показана модель сети, которую мы будем использовать для вывода алгоритма APEX. Как и ранее, входной вектор **x** имеет размерность m. Его компонентами являются $x_1, x_2, \ldots, x_m$. Все нейроны сети считаются линейными. Как показано на рис. 8.11, в этой сети существуют два типа синаптических весов.

- *Прямые связи* (feedforward connection) от входных узлов к каждому из нейронов 1, 2,..., j, где $j < m$. Здесь представляют интерес прямые связи, приходящие к нейрону j. Их можно представить в виде вектора прямых связей:

$$\mathbf{w}_j = [w_{j1}(n), w_{j2}(n), \ldots, w_{jm}(n)]^T.$$

  Прямые связи работают в соответствии с *правилом обучения Хебба* (Hebbian learning rule) и, таким образом, реализуют *самоусиление* (self-amplification).

---

[3] В [301] конфигурация нейронной сети, используемая для анализа главных компонентов, была расширена за счет включения Хеббоподобных обратных связей. Идея этой модификации была навеяна более ранней работой, посвященной адаптации и декорреляции в зрительной области коры головного мозга [93]. В этой работе было дано объясиение следующему факту. Если нейроны взаимодействуют согласно анти-Хеббовскому правилу, то выходы этих нейронов определяют систему координат, в которой нет корреляции, даже если входные сигналы сильно коррелированны.

Использование латерального торможения в выходных нейронах было также предложеио в [909], [910]. Однако, в отличие от модели, предложенной в [301], латеральные связи, описанные в этих статьях, не были симметричными. При этом латеральная сеть была иерархической в том смысле, что нейрон i тормозил все нейроны за исключением нейронов $1, 2, \ldots, i-1$, где $i = 1, 2, \ldots$.

Модель APEX, предложенная в [606], предполагает ту же топологию сети, что и модель, описанная в [909], [910], но для коррекции синаптических весов в прямых и латеральных связях сети используется другое правило обучения (см. раздел 8.4).

- *Латеральные связи* (lateral connection) между выходами нейронов $1, 2, \ldots, j-1$ и входом нейрона j. Они реализуют *обратные* связи в сети. Эти связи представляются вектором обратных связей:

$$\mathbf{a}_j(n) = [a_{j1}(n), a_{j2}(n), \ldots, a_{j,j-1}(n)]^T.$$

Латеральные связи действуют в соответствии с *анти-Хеббовским правилом обучения* (anti-Hebbian learning rule), что делает их *тормозящими* (inhibitory).

На рис. 8.11 прямые и латеральные связи нейрона j выделены жирным, чтобы подчеркнуть значение нейрона j как объекта изучения.

Выход $y_j(n)$ нейрона j можно представить в следующем виде:

$$y_j(n) = \mathbf{w}_j^T(n)\mathbf{x}(n) + \mathbf{a}_j^T(n)\mathbf{y}_{j-1}(n), \tag{8.98}$$

где слагаемое $\mathbf{w}_j^T(n)\mathbf{x}(n)$ отражает влияние прямых связей, а $\mathbf{a}_j^T(n)\mathbf{y}_{j-1}(n)$ — латеральных. Вектор сигнала обратной связи $\mathbf{y}_{j-1}(n)$ формируется выходами нейронов $1, 2, \ldots, j-1$:

$$\mathbf{y}_{j-1}(n) = [y_1(n), y_2(n), \ldots, y_{j-1}(n)]^T. \tag{8.99}$$

Предполагается, что входной вектор $\mathbf{x}(n)$ выбирается из стационарного процесса с матрицей корреляции $\mathbf{R}$, имеющей *различные собственные значения, отсортированные в порядке убывания*, т.е.

$$\lambda_1 > \lambda_2 > \ldots > \lambda_{j-1} > \lambda_j > \ldots > \lambda_m. \tag{8.100}$$

Далее предполагается, что нейроны $1, 2, \ldots, j-1$ сети, изображенной на рис. 8.11, *уже сошлись к своим устойчивым состояниям*:

$$\mathbf{w}_k(0) = \mathbf{q}_k, k = 1, 2, \ldots, j-1, \tag{8.101}$$

$$\mathbf{a}_k(0) = \mathbf{0}, k = 1, 2, \ldots, j-1, \tag{8.102}$$

где $\mathbf{q}_k$ — собственный вектор, связанный с k-м собственным значением матрицы корреляции $\mathbf{R}$, а момент времени n =0 соответствует началу вычислений в сети. Тогда, учитывая выражения (8.98), (8.99), (8.101) и (8.102), можно записать:

$$\mathbf{y}_{j-1}(n) = [\mathbf{q}_1^T\mathbf{x}(n), \mathbf{q}_2^T\mathbf{x}(n), \ldots, \mathbf{q}_{j-1}^T\mathbf{x}(n)] = \mathbf{Q}\mathbf{x}(n), \tag{8.103}$$

где $\mathbf{Q}$ — матрица размерности $(j-1) \times m$, определяемая в терминах собственных векторов $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{j-1}$, связанных с $(j-1)$ наибольшими собственными значениями $\lambda_1, \lambda_2, \ldots, \lambda_{j-1}$ матрицы корреляции $\mathbf{R}$, т.е.

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{j-1}]^T. \tag{8.104}$$

Нейрон j сети, показанной на рис. 8.11, требуется использовать для вычисления следующего наибольшего собственного числа λ_j матрицы корреляции $\mathbf{R}$ входного вектора $\mathbf{x}(n)$ и связанного с ним собственного вектора $\mathbf{q}_j$.

Уравнения коррекции для вектора прямых синаптических весов $\mathbf{w}_j(n)$ и вектора латеральных весов $\mathbf{a}_j(n)$ нейрона j имеют следующий вид:

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta \left[y_j(n)\mathbf{x}(n) - y_j^2(n)\mathbf{w}_j(n) \right] \tag{8.105}$$

и

$$\mathbf{a}_j(n+1) = \mathbf{a}_j(n) + \eta \left[y_j(n)\mathbf{y}_{j-1}(n) - y_j^2(n)\mathbf{a}_j(n) \right], \tag{8.106}$$

где η — *параметр скорости обучения* (learning-rate parameter), одинаковый в обоих уравнениях коррекции. Слагаемое $y_j(n)\mathbf{x}(n)$ в правой части уравнения (8.105) представляет обучение Хебба, в то время как слагаемое $-y_j(n)\mathbf{y}_{j-1}(n)$ в выражении (8.106) — анти-Хеббовское обучение. Оставшиеся слагаемые, $-y_j^2(n)\mathbf{w}_j(n)$ и $y_j^2(n)\mathbf{a}_j(n)$, включены в эти уравнения для обеспечения устойчивости алгоритма. В своей основе уравнение (8.105) представляет собой векторную форму правила обучения Ойа, описанного в (8.40), в то время как уравнение (8.106) является *совершенно новым* и учитывает латеральное торможение [258], [606].

Абсолютная устойчивость сети, изображенной на рис. 8.11, доказывается методом индукции.

- Сначала доказывается, что если нейроны 1, 2, ..., $j-1$ сходятся к своим устойчивым значениям, то нейрон j тоже сходится к своему устойчивому значению, выбирая следующее по счету наибольшее собственное значение λ_j матрицы корреляции $\mathbf{R}$ входного вектора $\mathbf{x}(n)$ и соответствующий ему собственный вектор $\mathbf{q}_j$.
- После этого мы завершаем доказательство методом индукции, учитывая, что нейрон 1 не имеет обратных связей и, таким образом, вектор весов обратной связи $\mathbf{a}_1$ является нулевым. Следовательно, этот конкретный нейрон функционирует таким же образом, как и нейрон Ойа, а из раздела 8.4 мы знаем, что этот нейрон при определенных условиях является абсолютно устойчивым.

Таким образом, пунктом, требующим наибольшего внимания, является первый.

Сделаем одно фундаментальное допущение, приведенное в разделе 8.4, и сформулируем следующую теорему в контексте нейрона j нейронной сети, показанной на рис. 8.11, работающей при выполнении условий (8.105) и (8.106) [258], [606].

Если параметр скорости обучения η принимает достаточно малое значение, обеспечивающее медленную корректировку весовых коэффициентов, то в пределе вектор прямых весов и усредненная мощность нейрона j (дисперсия) достигают нормализованного собственного вектора $\mathbf{q}_j$ и соответствующего собственного значения λ_j матрицы корреляции $\mathbf{R}$:

$$\lim_{n\to\infty} \mathbf{w}_j(n) = \mathbf{q}_j,$$
$$\lim_{n\to\infty} \sigma_j^2(n) = \lambda_j,$$

где $\sigma_j^2(n) = E[y_j^2(N)]$ и $\lambda_1 > \lambda_2 > \ldots > \lambda_j > \ldots > \lambda_m$. Другими словами, с помощью собственных векторов $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{j-1}$ нейрона j сети, показанной на рис. 8.11, можно вычислить наибольшее собственное значение λ_j и соответствующий ему собственный вектор $\mathbf{q}_j$.

Для доказательства этой теоремы рассмотрим уравнение (8.105). Используя выражения (8.98) и (8.99), а также свойство

$$\mathbf{a}_j^T(n)\mathbf{y}_{j-1}(n) = \mathbf{y}_{j-1}^T(n)\mathbf{a}_j(n),$$

выражение (8.105) можно переписать в следующем виде:

$$\begin{aligned}\mathbf{w}_j(n+1) =& \mathbf{w}_j(n) + \\ &+ \eta[\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}_j(n) + \mathbf{x}(n)\mathbf{x}^T(n)\mathbf{Q}^T\mathbf{a}_j(n) - y_j^2(n)\mathbf{w}_j(n)],\end{aligned} \tag{8.107}$$

где матрица $\mathbf{Q}$ определяется формулой (8.104). Слагаемое $y_j^2(n)$ оставлено без изменений, так как впоследствии оно будет отделено. При выполнении фундаментальных предположений, описанных в разделе 8.4, оказывается, что применение оператора статистического ожидания к обеим частям равенства (8.107) приводит к следующему:

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta[\mathbf{R}\mathbf{w}_j(n) + \mathbf{R}\mathbf{Q}^T\mathbf{a}_j(n) - \sigma_j^2(n)\mathbf{w}_j(n)], \tag{8.108}$$

где $\mathbf{R}$ — матрица корреляции входного вектора $\mathbf{x}(n)$; $\sigma_j^2(n)$ — усредненная выходная мощность вектора j. Пусть вектор синаптических весов $\mathbf{w}_j(n)$ является расширенным в терминах полного ортогонального множества собственных векторов матрицы корреляции $\mathbf{R}$:

$$\mathbf{w}_j(n) = \sum_{k=1}^{m} \theta_{jk}(n)\mathbf{q}_k, \tag{8.109}$$

где $\mathbf{q}_k$ — собственный вектор, связанный с собственным значением λ_k матрицы $\mathbf{R}$; $\theta_{jk}(n)$ — коэффициент разложения, зависящий от времени. Тогда можно использовать основное соотношение (см. (8.14))

$$\mathbf{R}\mathbf{q}_k = \lambda_k \mathbf{q}_k,$$

чтобы выразить матричное произведение $\mathbf{R}\mathbf{w}_j(n)$ следующим образом:

$$\mathbf{R}\mathbf{w}_j(n) = \sum_{k=1}^{m} \theta_{jk}(n)\mathbf{R}\mathbf{q}_k = \sum_{k=1}^{m} \lambda_k \theta_{jk}(n)\mathbf{q}_k. \tag{8.110}$$

Аналогично, используя выражение (8.104), матричное произведение $\mathbf{R}\mathbf{Q}^T\mathbf{a}_j(n)$ можно представить в виде

$$\begin{aligned}\mathbf{R}\mathbf{Q}^T\mathbf{a}_j(n) &= \mathbf{R}[\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{j-1}]\mathbf{a}_j(n) = \\ &= [\lambda_1\mathbf{q}_1, \lambda_2\mathbf{q}_2, \ldots, \lambda_{j-1}\mathbf{q}_{j-1}] \begin{bmatrix} a_{j1}(n) \\ a_{j2}(n) \\ \ldots \\ a_{j,j-1}(n) \end{bmatrix} = \sum_{k=1}^{j-1} \lambda_k a_{jk}(n)\mathbf{q}_k.\end{aligned} \tag{8.111}$$

Тогда, подставляя выражения (8.109)–(8.111) в равенство (8.108), получим [606]:

$$\sum_{k=1}^{m} \theta_{jk}(n+1)\mathbf{q}_k = \sum_{k=1}^{m} \{1 + \eta[\lambda_k - \sigma_j^2(n)]\}\theta_{jk}(n)\mathbf{q}_k + \eta \sum_{k=1}^{j-1} \lambda_k a_{jk}(n)\mathbf{q}_k. \tag{8.112}$$

С помощью описанной выше процедуры можно показать, что уравнение коррекции (8.106) для вектора весов обратной связи $\mathbf{a}_j(n)$ может быть преобразовано к виду (см. задачу 8.7)

$$\mathbf{a}_j(n+1) = -\eta\lambda_k\theta_{jk}(n)\mathbf{1}_k + \{1 - \eta[\lambda_k + \sigma_j^2(n)]\}\mathbf{a}_j(n), \tag{8.113}$$

где $\mathbf{1}_k$ — вектор, все элементы которого равны нулю, за исключением k-го, равного единице. Индекс k при этом находится в диапазоне от 1 до $(j-1)$.

Следует рассмотреть два случая, в зависимости от положения индекса k по отношению к значению $j-1$. В первом случае он находится слева от границы, т.е. $1 \le k \le j-1$. Этот случай относится к анализу "старых" главных мод сети. Во втором случае индекс k находится справа от границы, т.е. $j \le k \le m$. Этот случай относится к анализу "новых" главных мод. Общее количество главных мод равно m — размерности входного вектора $\mathbf{x}(n)$.

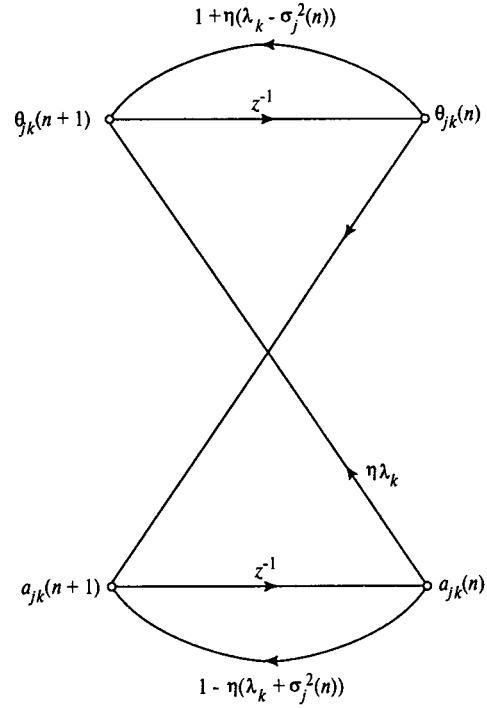


Рис. 8.12. Граф передачи сигнала для выражений (8.114) и (8.115)

Случай 1. $1 \leq k \leq j-1$

Для данного случая выведем следующие уравнения коррекции для коэффициента $\theta_{jk}(n)$, связанного с собственным вектором $\mathbf{q}_k$ и весом обратной связи $a_{jk}(n)$ из выражений (8.112) и (8.113) соответственно:

$$\theta_{jk}(n+1) = -\eta\lambda_k a_{jk}(n) + \{1 + \eta[\lambda_k - \sigma_j^2(n)]\}\theta_{jk}(n) \tag{8.114}$$

и

$$a_{jk}(n+1) = -\eta\lambda_k \theta_{jk}(n) + \{1 - \eta[\lambda_k + \sigma_j^2(n)]\}a_{jk}(n). \tag{8.115}$$

На рис. 8.12 представлен граф передачи сигнала для выражений (8.114) и (8.115). Выражения (8.114) и (8.115) можно переписать в следующем матричном виде:

$$\begin{bmatrix} \theta_{jk}(n+1) \\ a_{jk}(n+1) \end{bmatrix} = \begin{bmatrix} 1 + \eta[\lambda_k - \sigma_j^2(n)] & \eta\lambda_k \\ -\eta\lambda_k & 1 - \eta[\lambda_k + \sigma_j^2(n)] \end{bmatrix} \begin{bmatrix} \theta_{jk}(n) \\ a_{jk}(n) \end{bmatrix}. \tag{8.116}$$

Система, описанная выражением (8.116), имеет двойное собственное значение:

$$\rho_{jk} = [1 - \eta\sigma_j^2(n)]^2. \tag{8.117}$$

Из выражения (8.117) можно сделать два важных вывода.

1. Двойное собственное значение ρ_{jk} матричной системы (8.116) является независимым от всех собственных значений λ_k матрицы корреляции $\mathbf{R}$, соответствующих $k = 1, 2, \ldots, j-1$.
2. Для всех k двойное собственное значение ρ_{jk} зависит только от параметра скорости обучения η и средней выходной мощности σ_j^2 нейрона j. Следовательно, это значение будет меньше единицы, поскольку η — достаточно малое положительное число.

Если $\rho_{jk} < 1$, то коэффициенты разложения $\theta_{jk}(n)$ в (8.109) и веса обратной связи $a_{jk}(n)$ для всех k будут асимптотически стремиться к нулю с одинаковой скоростью, так как все основные моды сети имеют одно и то же собственное значение [258], [606]. Результатом является то свойство, что ортогональность собственных векторов матрицы корреляции не зависит от собственных чисел. Другими словами, разложение $\mathbf{w}_j(n)$ в терминах ортогонального множества собственных векторов матрицы корреляции $\mathbf{R}$, представленное в (8.109) и положенное в основу формулы (8.117), *инвариантно* к выбору собственных значений $\lambda_1, \lambda_2, \ldots, \lambda_{j-1}$.

Случай 2. $j \leq k \leq m$

Во втором случае веса обратной связи $a_{jk}(n)$ *не влияют* на моды сети, т.е.

$$a_{jk}(n) = 0 j \leq k \leq m. \tag{8.118}$$

Следовательно, для любой основной моды $k \geq j$ выполняется следующее простое равенство:

$$\theta_{jk}(n+1) = \{1 + \eta[\lambda_k - \sigma_j^2(n)]\}\theta_{jk}(n), \tag{8.119}$$

которое непосредственно следует из равенств (8.112) и (8.118). Согласно случаю 1, векторы $a_{jk}(n)\theta_{jk}(n)$ сходятся к нулю при $k = 1, 2, \ldots, j$-1. Если выходом линейного нейрона j является случайная переменная $Y_j(n)$, ее среднюю дисперсию можно выразить так:

$$\sigma_j^2(n) = E[Y_j^2(n)] = \sum_{k=j}^{m} \lambda_k \theta_{jk}^2(n), \tag{8.120}$$

где в последней строке использовалось соотношение

$$\mathbf{q}_k^T \mathbf{R} \mathbf{q}_l = \begin{cases} \lambda_k, & l = k, \\ 0 - \text{в противном случае.} \end{cases}$$

Отсюда следует, что уравнение (8.119) не может расходиться, поскольку с ростом $\theta_{jk}(n)$, при котором $\sigma_j^2(n) > \lambda_k$, выражение $1 + \eta[\lambda_k - \sigma_j^2(n)]$ становится меньше единицы, и в этом случае $\theta_{jk}(n)$ убывает по амплитуде. Пусть алгоритм инициализируется такими значениями, что $\theta_{jj}(0) \neq 0$. Введем следующее определение:

$$r_{jk}(n) = \frac{\theta_{jk}(n)}{\theta_{jj}(n)}, \quad k = j+1, \ldots, m. \tag{8.121}$$

Тогда, используя (8.119), можно записать:

$$r_{jk}(n+1) = \frac{1 + \eta[\lambda_k - \sigma_j^2(n)]}{1 + \eta[\lambda_j - \sigma_j^2(n)]} r_{jk}(n). \tag{8.122}$$

Если собственные числа матрицы корреляции упорядочены по убыванию

$$\lambda_1 > \lambda_2 > \ldots > \lambda_k > \ldots > \lambda_j > \ldots > \lambda_m,$$

то

$$\frac{\theta_{jk}(n)}{\theta_{jj}(n)} < 1 \text{ для всех } n \text{ и для } \quad k = j+1, \ldots, m. \tag{8.123}$$

Более того, из выражений (8.119) и (8.120) видно, что значение $\theta_{jj}(n+1)$ остается ограниченным, следовательно:

$$r_{jk}(n) \to 0 \text{ при } n \to \infty \text{ для } k = j+1, \ldots, m. \tag{8.124}$$

Аналогично, в свете определения (8.121) можно утверждать, что

$$\theta_{jk}(n) \to 0 \text{ при } n \to \infty \text{ для } k = j+1, \ldots, m. \tag{8.125}$$

При этом условии выражение (8.120) сводится к следующему:

$$\sigma_j^2(n) = \lambda_j \theta_{jj}^2(n), \tag{8.126}$$

и, следовательно, равенство (8.119) для $k = j$ принимает вид

$$\theta_{jj}(n+1) = \{1 + \eta\lambda_j[1 - \theta_{jj}(n)]\}\theta_{jj}(n). \tag{8.127}$$

Из этого соотношения можно получить, что

$$\theta_{jj}(n) \to 1 \text{ при } n \to \infty. \tag{8.128}$$

Из данного предельного условия и выражения (8.125) можно сделать два следующих вывода.

1. Из выражения (8.126) получим

$$\sigma_j^2(n) \to \lambda_j \text{ при } n \to \infty. \tag{8.129}$$

2. Из выражения (8.109) получим

$$\mathbf{w}_j(n) \to \mathbf{q}_j \text{ при } n \to \infty. \tag{8.130}$$

Другими словами, модель нейронной сети, показанная на рис. 8.11, извлекает j-е собственное значение и связанный с ним собственный вектор матрицы корреляции $\mathbf{R}$ входного вектора $\mathbf{x}(n)$, если количество итераций n стремится к бесконечности. При этом, естественно, предполагается, что нейроны $1, 2, \ldots, j-1$ сети уже сошлись к соответствующим собственным значениям и собственным векторам матрицы корреляции $\mathbf{R}$.

Представленная здесь трактовка алгоритма APEX предполагает, что перед тем, как нейрон j вступает в действие, нейроны 1, 2,..., $j-1$ сети уже обучены. Это сделано для того, чтобы упростить изложение материала. Однако на практике нейроны в алгоритме APEX обучаются совместно[4].

Интенсивность обучения

В алгоритме APEX, описываемом уравнениями (8.105) и (8.106), для коррекции вектора прямых весов $\mathbf{w}_j(n)$ и вектора весов обратной связи $\mathbf{a}_j(n)$ используется один и тот же параметр скорости обучения η. Для определения оптимального значения параметра η для каждого нейрона j можно использовать соотношение (8.117), установив двойное собственное значение ρ_{jk} в значение нуль. В этом случае получим:

$$\eta_{j,\text{opt}}(n) = \frac{1}{\sigma_j^2(n)}, \tag{8.131}$$

где $\sigma_j^2(n)$ — средняя мощность выхода нейрона j. Однако более практичным будет

[4] Строгое доказательство сходимости алгоритма APEX, при которой все нейроны обучаются совместно, было предложено в [182].

следующее значение этого параметра [258], [606]:

$$\eta_j = \frac{1}{\lambda_{j-1}}. \tag{8.132}$$

При таком значении параметра скорости обучения $\lambda_{j-1} > \lambda_j$ и $\sigma_j^2(n) \to \lambda_j$ при $n \to \infty$. Обратите внимание, что собственное значение λ_{j-1} вычисляется нейроном $(j-1)$ и, таким образом, доступно для использования при коррекции прямых и латеральных весов нейрона j.

Алгоритм APEX в сжатом виде

1. Инициализируем вектор прямых весов $\mathbf{w}_j$ и вектор весов обратных связей $\mathbf{a}_j$ малыми случайными значениями для момента времени $n = 1$, $j = 1, 2, \ldots, m$. Параметру скорости обучения η присвоим некоторое малое положительное число.
2. Пусть $j = 1$; тогда для $n = 1, 2, \ldots$ вычислим:

$$\begin{aligned} y_1(n) &= \mathbf{w}_1^T(n)\mathbf{x}(n), \\ \mathbf{w}_1(n+1) &= \mathbf{w}_1(n) + \eta[y_1(n)\mathbf{x}(n) - y_1^2(n)\mathbf{w}_1(n)], \end{aligned}$$

   где $\mathbf{x}(n)$ — входной вектор. Для больших n $\mathbf{w}_1(n) \to \mathbf{q}_1$, где $\mathbf{q}_1$ — собственный вектор, связанный с наибольшим собственным значением λ_1 матрицы корреляции вектора $\mathbf{x}(n)$.
3. Принимаем j =2 и для $n = 1, 2, \ldots$ вычислим:

$$\begin{aligned} \mathbf{y}_{j-1}(n) &= [y_1(n), y_2(n), \ldots, y_{j-1}(n)]^T, \\ \mathbf{y}_j(n) &= \mathbf{w}_j^T(n)\mathbf{x}(n) + \mathbf{a}_j^T(n)\mathbf{y}_{j-1}(n), \\ \mathbf{w}_j(n+1) &= \mathbf{w}_j(n) + \eta[y_j(n)\mathbf{x}(n) - y_j^2(n)\mathbf{w}_j(n)], \\ \mathbf{a}_j(n+1) &= \mathbf{a}_j(n) - \eta[y_j(n)\mathbf{y}_{j-1}(n) - y_j^2(n)\mathbf{a}_j(n)]. \end{aligned}$$

4. Увеличиваем j на единицу, переходим к шагу 3 и продолжаем до $j = m$, где m — нужное количество главных компонентов. (Обратите внимание, что $j = 1$ соответствует собственному вектору, связанному с наибольшим собственным значением, которое вычисляется на шаге 2.) Для больших $n\mathbf{w}_j(n) \to \mathbf{q}_j$ и $\mathbf{a}_j(n) \to \mathbf{0}$, где $\mathbf{q}_j$ — собственный вектор, ассоциированный с j-м собственным значением λ_j матрицы корреляции вектора $\mathbf{x}(n)$.

8.8. Два класса алгоритмов PCA

В дополнение к обобщенному алгоритму Хебба (GHA), который рассматривался в разделе 8.5, и алгоритму APEX, описанному в разделе 8.7, существует и ряд других алгоритмов анализа главных компонентов (PCA), широко описанных в литературе[5]. Различные алгоритмы PCA на основе нейронных сетей можно разбить на два класса: *алгоритмы повторного оценивания* (reestimation algorithm) и *алгоритмы декорреляции* (decorrelating algorithm).

Согласно этой классификации, GHA является алгоритмом повторного оценивания, так как выражения (8.87) и (8.88) можно записать в эквивалентном виде:

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta y_j(n)[\mathbf{x}(n) - \hat{\mathbf{x}}_j(n), \tag{8.133}$$

где $\hat{\mathbf{x}}_j(n)$ — *операция повторной оценки* (reestimator), которая определяется следующим образом:

$$\hat{\mathbf{x}}_j(n) = \sum_{k=1}^{j} \mathbf{w}_k(n) y_k(n). \tag{8.134}$$

В алгоритме повторной оценки нейронная сеть имеет только прямые связи, и ее веса корректируются согласно алгоритму Хебба. Последующие выходы сети должны обучаться другим главным компонентам, при этом оценки предшествующих компонентов вычитаются из входного сигнала перед тем, как эти данные поступят для использования в процессе обучения.

В отличие от этого алгоритма APEX является алгоритмом декорреляции. В таких алгоритмах нейронная сеть имеет как прямые, так и обратные связи. Работа прямых связей основана на правиле обучения Хебба, в то время как работа обратных — на анти-Хеббовском правиле. Последующие выходы сети декоррелируются, заставляя сеть воспроизводить разные главные компоненты.

Подпространство главных компонентов

В ситуациях, где используется только *подпространство главных компонентов* (principal subspace), можно применять *симметричную модель* (symmetric model), в которой оценка $\hat{\mathbf{x}}_j(n)$ в алгоритме GHA заменена на следующую:

$$\hat{\mathbf{x}}(n) = \sum_{k=1}^{l} \mathbf{w}_k(n) y_k(n) \text{ для всех } l. \tag{8.135}$$

---

[5] Обсуждение разных нейронных моделей, используемых для анализа главных компонентов, и их сравнение предлагается в [258].

В симметричной модели, определяемой формулами (8.133) и (8.135), сеть сходится к множеству выходов, которые определяют подпространство главных компонентов, а не к самим основным компонентам. При сходимости векторы весов этой сети ортогональны друг другу, как и в GHA. Описанное здесь подпространство главных компонентов можно рассматривать как обобщение классического правила Ойа (Oja), определяемого соотношением (8.46).

8.9. Пакетный и адаптивный методы вычислений

Обсуждение анализа главных компонентов будет не полным, если не рассмотреть вычислительные аспекты этой задачи. В этом контексте можно сказать, что существуют два основных подхода к вычислению главных компонентов: пакетный и адаптивный.

Метод декомпозиции на основе *собственных векторов* (eigendecomposition), описанный в разделе 8.3, и связанный с ним метод сингулярного разложения принадлежат к категории *пакетных*. С другой стороны, алгоритмы GHA и APEX (см. разделы 8.5 и 8.7) принадлежат к категории *адаптивных*.

Теоретически сингулярная декомпозиция основана на усредненной по множеству матрице корреляции $\mathbf{R}$ случайного вектора $\mathbf{X}(n)$ (см. раздел 8.3). На практике же мы используем оценку матрицы корреляции $\mathbf{R}$. Пусть $\{\mathbf{x}(n)\}_{n=1}^{N}$ — множество N реализаций случайного вектора $\mathbf{X}(n)$ в равномерно распределенные дискретные моменты времени. Имея такое множество наблюдений, в качестве оценки матрицы корреляции можно использовать *простое среднее*:

$$\hat{\mathbf{R}}(N) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}(n)\mathbf{x}^{T}(n). \tag{8.136}$$

Предполагая, что среда, представленная случайным вектором $\mathbf{X}(n)$, является *эргодической* (ergodic), простое среднее $\hat{\mathbf{R}}(N)$ достигает значения $\mathbf{R}$ при достижении размером множества N бесконечности. На этом основании к простому среднему $\hat{\mathbf{R}}(N)$ можно применить процедуру разложения по собственным векторам, вычислить его собственные значения и ассоциированные с ними собственные векторы, используя вместо матрицы $\mathbf{R}$ матрицу оценки $\hat{\mathbf{R}}(N)$.

Однако с точки зрения вычислений самым лучшим методом является *сингулярная декомпозиция* (singular value decomposition — SVD), применяемая непосредственно к *матрице данных*. Для множества наблюдений $\{\mathbf{x}(n)\}_{n=1}^{N}$ матрица данных определяется следующим образом:

$$\mathbf{A} = [\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N)]. \tag{8.137}$$

Тогда несложно заметить, что если не учитывать масштабирующий множитель $1/N$, то оценка $\hat{\mathbf{R}}(N)$ матрицы корреляции $\mathbf{R}$ равна скалярному произведению $\mathbf{A}\mathbf{A}^T$. Согласно *теореме о сингулярной декомпозиции* (см. главу 5), матрица данных $\mathbf{A}(n)$ может быть представлена в виде декомпозиции [368]:

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T, \tag{8.138}$$

где $\mathbf{U}$ и $\mathbf{V}$ — ортогональные матрицы, т.е.

$$\mathbf{U}^{-1} = \mathbf{U}^T \tag{8.139}$$

и

$$\mathbf{V}^{-1} = \mathbf{V}^T. \tag{8.140}$$

Матрица $\boldsymbol{\Sigma}$ имеет следующую структуру:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \begin{bmatrix} \sigma_1 & & \mathbf{0} \\ & \sigma_2 & \\ & & \ddots \\ \mathbf{0} & & \sigma_k \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{8.141}$$

где $k \leq m$; m — размерность вектора наблюдений $\mathbf{x}(n)$. Числа $\sigma_1, \sigma_2, \ldots, \sigma_k$ называются *сингулярными значениями* (singular value) матрицы данных $\mathbf{A}$. Соответственно столбцы ортогональной матрицы $\mathbf{U}$ называются *левыми сингулярными векторами* (left singular vector), а столбцы матрицы $\mathbf{V}$ — *правыми сингулярными векторами*. Сингулярная декомпозиция матрицы данных $\mathbf{A}$ связана с декомпозицией оценки $\hat{\mathbf{R}}(N)$ матрицы корреляции по собственным векторам в следующих аспектах.

- За исключением масштабирующего множителя $1/\sqrt{N}$ сингулярные значения матрицы данных $\mathbf{A}$ являются квадратным корнем из собственных значений оценки $\hat{\mathbf{R}}(N)$.
- Левые сингулярные векторы матрицы данных $\mathbf{A}$ являются собственными векторами матрицы оценки $\hat{\mathbf{R}}(N)$.

Теперь вычислительные преимущества сингулярной декомпозиции становятся очевидными. Для заранее заданной точности вычислений процедура сингулярной декомпозиции требует вдвое меньшей вычислительной мощности, чем процедура разложения по собственным векторам. Более того, для компьютерной реализации процедуры сингулярной декомпозиции существует множество алгоритмов и высокоточных программ [368], [434], [435]. Однако ввиду жестких требований к хранению

данных на практике использование этих программ может быть ограничено не слишком большими размерами множеств.

Переходя к категории *адаптивных методов*, следует сказать, что они работают с множествами произвольных размеров. Для всех практических реализаций не существует каких-либо ограничений на N. Адаптивные методы можно проиллюстрировать на примере Хеббовских нейронных сетей, работа которых основана на идеях из нейробиологии. Требования к хранению данных в таких методах являются относительно умеренными, так как в них не нужно хранить промежуточные значения собственных значений и собственных векторов. Еще одной привлекательной чертой адаптивных алгоритмов является то, что в *нестационарной среде* (nonstationary environment) они имеют встроенную способность *отслеживать* постепенные изменения в оптимальном решении незатратным способом (по сравнению с пакетными методами). При этом основным недостатком адаптивных алгоритмов типа стохастической аппроксимации является их относительно малая скорость сходимости. Это особенно отчетливо наблюдается в больших стационарных задачах, даже при реализации адаптивных методов на параллельных аппаратных нейронных сетях [594].

8.10. Анализ главных компонентов на основе ядра

Форма PCA, которую мы рассматривали до сих пор, подразумевает вычисления в пространстве входных данных. Теперь рассмотрим другую форму алгоритмов PCA, в которой вычисления осуществляются в пространстве признаков, являющемся *нелинейным отображением* входного пространства. Пространство признаков, в соответствии с теоремой Мерсера, определяется *ядром скалярного произведения* (inner-product kernel). Вопросы, связанные с ядром скалярного произведения, рассматривались в главе 6, посвященной машинам опорных векторов. Идея *анализа главных компонентов на основе ядра* (kernel-based principal components analysis) была предложена в [947].

Ввиду нелинейности отображения входного пространства в пространство признаков ядро PCA также является нелинейным. Однако, в отличие от других форм нелинейных PCA[6], реализация PCA на основе ядра базируется на линейной алгебре. Таким образом, PCA на основе ядра можно рассматривать как естественное расширение обычных PCA.

---

[6] Методы нелинейного анализа главных компонентов (за исключением PCA ядра) можно разделить на три класса [258].

Сети Хебба (Hebbian networks), получаемые путем замены линейных нейронов в подобных Хеббовским алгоритмах PCA нелинейными нейронами [542].

Сети репликации (replicator network) или *системы автокодирования* (autoencoder), созданные на базе многослойных персептронов (о сетях репликации см. в главе 4).

Главные кривые (principal curve), основанные на итеративной оценке кривой или поверхности, описывающей структуру данных [430]. В [186] и [891] указывалось, что самоорганизующиеся карты Кохонена можно рассматривать как вычислительную процедуру поиска дискретной аппроксимации главных кривых. Самоорганизующиеся карты рассматриваются в следующей главе.

Пусть вектор $\varphi(\mathbf{x}_j)$ определяет образ входного вектора $\mathbf{x}_j$, индуцированный в пространстве признаков, определяемом нелинейным отображением $j : \Re^{m_0} \rightarrow \Re^{m_1}$, где m_0 — размерность входного пространства; m_1 — размерность пространства признаков. Для множества примеров $\{\mathbf{x}_i\}_{i=1}^N$ получаем соответствующее множество векторов признаков $\{\varphi(\mathbf{x}_i)\}_{i=1}^N$. Соответственно в пространстве признаков можно определить матрицу корреляции $\tilde{\mathbf{R}}$ размерности $m_1 \times m_1$ следующего вида:

$$\tilde{\mathbf{R}} = \frac{1}{N}\sum_{i=1}^{N} \varphi(\mathbf{x}_i)\varphi^T(\mathbf{x}_i). \tag{8.142}$$

Как и в обычном PCA, первое, что нужно сделать, — это убедиться, что множество векторов признаков $\{\varphi(\mathbf{x}_i)\}_{i=1}^N$ имеет нулевое среднее значение:

$$\frac{1}{N}\sum_{i=1}^{N} \varphi(\mathbf{x}_i) = \mathbf{0}.$$

Выполнение этого условия в пространстве признаков является более сложным требованием, чем его выполнение во входном пространстве. В задаче 8.10 будет описана процедура для выполнения этого требования. Далее будем предполагать, что векторы признаков уже отцентрированы. Тогда выражение (8.14) можно адаптировать к нашей ситуации, записав следующее:

$$\tilde{\mathbf{R}}\tilde{q} = \tilde{\lambda}\tilde{\mathbf{q}}, \tag{8.143}$$

где $\tilde{\lambda}$ — собственное значение матрицы корреляции $\tilde{\mathbf{R}}$; $\tilde{\mathbf{q}}$ — ассоциированный с ним собственный вектор. Далее заметим, что все собственные векторы, которые удовлетворяют равенству (8.143) при $\tilde{\lambda} \neq 0$, лежат в линейной оболочке множества векторов признаков $\{\varphi(\mathbf{x}_j)\}_{j=1}^N$. Следовательно, существует такое множество коэффициентов $\{\alpha_j\}_{j=1}^N$, для которых

$$\tilde{\mathbf{q}} = \sum_{j=1}^{N} \alpha_j \varphi(\mathbf{x}_j). \tag{8.144}$$

Подставляя (8.142) и (8.144) в (8.143), получим:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_j \varphi(\mathbf{x}_i) K(\mathbf{x}_i, \mathbf{x}_j) = N\tilde{\lambda}\sum_{j=1}^{N} \alpha_j \varphi(\mathbf{x}_i), \tag{8.145}$$

где $K(\mathbf{x}_i, \mathbf{x}_j)$ — *ядро скалярного произведения* (inner-product kernel), определяемое в терминах векторов признаков как

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}_j). \tag{8.146}$$

Теперь сделаем еще один шаг и выразим уравнение (8.145) только в терминах ядра скалярного произведения. Для этого умножим обе части (8.145) на транспонированный вектор $\varphi^T(\mathbf{x}_k)$, в результате чего получим:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_j K(\mathbf{x}_k,\mathbf{x}_j)K(\mathbf{x}_i,\mathbf{x}_j) = N\tilde{\lambda}\sum_{j=1}^{N}\alpha_j K(\mathbf{x}_k,\mathbf{x}_j), \quad k=1,2,\ldots,N, \tag{8.147}$$

где ядра $K(\mathbf{x}_k, \mathbf{x}_i)$ и $K(\mathbf{x}_k, \mathbf{x}_j)$ сформированы согласно (8.146).

Теперь введем два матричных определения.

- Матрицу $\mathbf{K}$ размерности $N \times N$ назовем *матрицей ядра*. Ее ij-й элемент является ядром скалярного произведения $K(\mathbf{x}_i, \mathbf{x}_j)$.
- Введем вектор $\boldsymbol{\alpha}$ размерности $N \times 1$, j-ми элементами которого являются коэффициенты α_j.

Тогда выражение (8.147) можно переписать в более компактном, матричном виде

$$\mathbf{K}^2\boldsymbol{\alpha} = N\tilde{\lambda}\mathbf{K}\boldsymbol{\alpha}, \tag{8.148}$$

где под квадратом матрицы $\mathbf{K}^2$ понимается произведение матрицы $\mathbf{K}$ саму на себя. Так как эта матрица встречается в обеих частях уравнения (8.148), все решения задачи поиска собственных значений сведутся к решению упрощенной задачи

$$\mathbf{K}\boldsymbol{\alpha} = N\tilde{\lambda}\boldsymbol{\alpha}. \tag{8.149}$$

Пусть $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ — собственные значения матрицы ядра $\mathbf{K}$, т.е.

$$\lambda_j = N\tilde{\lambda}_j, \quad j=1,2,\ldots,N, \tag{8.150}$$

где $\tilde{\lambda}_j$ — j-е собственное значение матрицы корреляции $\tilde{\mathbf{R}}$. Тогда выражение (8.149) принимает стандартный вид

$$\mathbf{K}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}, \tag{8.151}$$

где вектор коэффициентов $\boldsymbol{\alpha}$ выступает в роли собственного вектора, связанного с собственным значением λ матрицы ядра $\mathbf{K}$. Вектор $\boldsymbol{\alpha}$ строится с учетом требования того, чтобы собственный вектор $\tilde{\mathbf{q}}$ матрицы $\tilde{\mathbf{R}}$ был нормализованным (т.е. имел единичную длину):

$$\tilde{\mathbf{q}}_k^T\tilde{\mathbf{q}}_k = 1 \text{ для } k=1,2,\ldots,p, \tag{8.152}$$

где предполагается, что собственные числа упорядочены по убыванию, а λ_p – наименьшее ненулевое собственное значение матрицы ядра $\mathbf{K}$. Используя (8.144) и (8.151), можно показать, что условие нормализации (8.152) эквивалентно следующему:

$$\boldsymbol{\alpha}_k^T \boldsymbol{\alpha}_k = \frac{1}{\lambda_k}, \quad k = 1, 2, \ldots, p. \tag{8.153}$$

Для извлечения главных компонентов нужно вычислить проекции на собственные векторы $\tilde{\mathbf{q}}_k$ пространства признаков:

$$\tilde{\mathbf{q}}_k^T \varphi(\mathbf{x}) = \sum_{j=1}^{N} \alpha_{k,j} \varphi^T(\mathbf{x}_j) \varphi(\mathbf{x}) = \sum_{j=1}^{N} \alpha_{k,j} K(\mathbf{x}_j, \mathbf{x}), \quad k = 1, 2, \ldots, p, \tag{8.154}$$

где вектор $\mathbf{x}$ — "тестовая" точка; $\alpha_{k,j}$ — j-й коэффициент собственного вектора α_k, связанного с k-м собственным значением матрицы $\mathbf{K}$. Проекция (8.154) определяет *нелинейные главные компоненты* (nonlinear principal component) m_1-мерного пространства признаков.

На рис. 8.13 проиллюстрирована основная идея алгоритма PCA на основе ядра, в которой пространство признаков нелинейно связано с входным пространством посредством преобразования $\varphi(\mathbf{x})$. Части (а) и (б) на рисунке обозначают входное пространство и пространство признаков соответственно. Контурная линия на рис. 8.13, *б* представляет собой постоянную проекцию на главный собственный вектор, показанный на рисунке символом вектора. На этом рисунке предполагается, что преобразование $\varphi(\mathbf{x})$ выбрано так, что образы точек данных, индуцированные в пространстве признаков, вытягиваются вдоль собственного вектора. На рис. 8.13, *а* показаны *нелинейные* контурные линии во входном пространстве, которые соответствуют прямым в пространстве признаков. Заметим, что мы преднамеренно не показали прообраз собственного вектора во входном пространстве, так как он может не существовать [947].

Для ядра скалярного произведения, определенного в соответствии с теоремой Мерсера, выполняем обычный алгоритм PCA в m_1-мерном пространстве признаков, где m_1 — параметр. Все свойства обычного алгоритма PCA, описанные в разделе 8.3, сохраняются и для PCA на основе ядра. В частности, PCA на основе ядра является линейным в пространстве признаков, но нелинейным во входном пространстве. Как таковой он может быть применен ко всем тем областям, где обычный алгоритм PCA использовался для извлечения признаков или сокращения размерности данных, при котором нелинейное расширение имеет смысл.

В главе 6 были представлены три метода построения ядра скалярного произведения, основанные на использовании полиномов, радиально-базисных и гиперболических функций (см. табл. 6.1). Открытым остался вопрос о том, как выбрать наиболее подходящее ядро для конкретной задачи (т.е. соответствующего пространства признаков) [948].

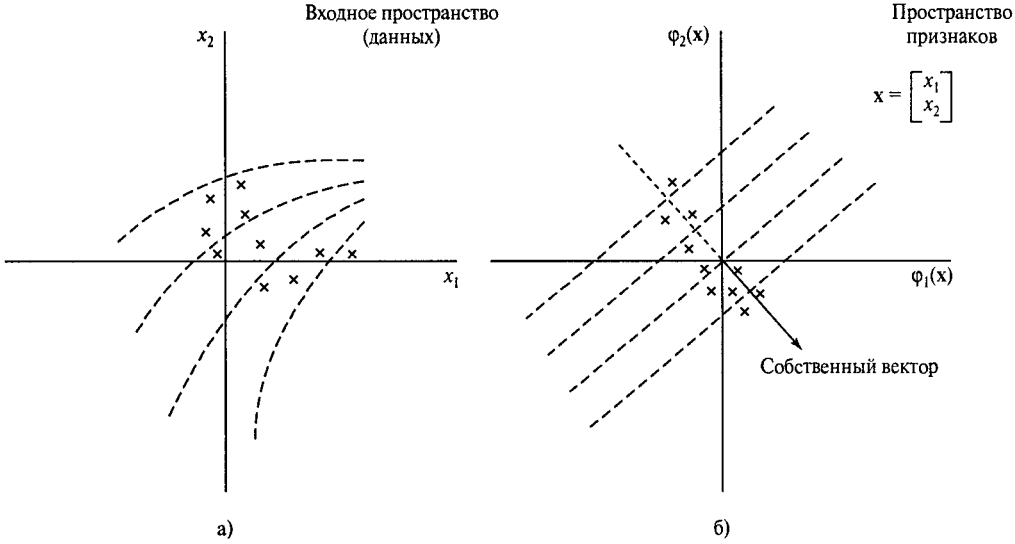


Рис. 8.13. Алгоритм PCA на основе ядра. Двумерное входное пространство с множеством точек данных (а); двумерное пространство признаков с образами точек части (а), вытянутыми вдоль собственного вектора (б). Равномерно распределенные пунктирные прямые в части (б) представляют контуры константной проекции на собственный вектор; соответствующие контуры во входном пространстве являются нелинейными

Алгоритм PCA на основе ядра в сжатом виде

1. Для данного множества примеров $\{\mathbf{x}_i\}_{i=1}^N$ вычислим матрицу ядра $\mathbf{K}=\{K(\mathbf{x}_i, \mathbf{x}_j)\}$, где

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}_j).$$

2. Решаем задачу на собственные значения:

$$\mathbf{K}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha},$$

где λ — собственное значение матрицы $\mathbf{K}$; $\boldsymbol{\alpha}$ — соответствующий ему собственный вектор.

3. Нормализуем собственные векторы, применяя требование

$$\boldsymbol{\alpha}_k^T\boldsymbol{\alpha}_k = \frac{1}{\lambda_k}, k = 1, 2, \ldots, p, \tag{8.155}$$

где λ_p — наименьшее ненулевое собственное значение матрицы $\mathbf{K}$ (в предположении, что собственные значения отсортированы по убыванию).

4. Для извлечения главных компонентов тестовой точки $\mathbf{x}$ вычислим проекции

$$\alpha_k = \tilde{\mathbf{q}}_k^T \varphi(\mathbf{x}) = \sum_{j=1}^{N} \alpha_{k,j} K(\mathbf{x}_j, \mathbf{x}),\ k = 1, 2, \ldots, p,$$

где $\alpha_{k,j}$ — j-й коэффициент собственного вектора α_k.

Пример 8.3

Для того чтобы лучше разобраться в работе алгоритма PCA на основе ядра, на рис. 8.14 показаны результаты простого эксперимента, описанного в [947]. Двумерные данные, состоящие из компонентов x_1 и x_2 и используемые в этом эксперименте, можно описать следующим образом: x_1-значения имеют равномерное распределение в интервале $[-1, 1]$; x_2-значения нелинейно связаны с элементами x_1 следующей формулой:

$$x_2 = x_1^2 + v,$$

где v — аддитивный гауссов шум с нулевым средним значением и дисперсией, равной 0,04.

Результаты PCA, показанные на рис. 8.14, были получены с помощью полиномов ядра:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}^T \mathbf{x}_i)^d, d = 1, 2, 3, 4,$$

где $d = 1$ соответствует линейному PCA, а $d = 2, 3, 4$ — PCA на основе ядра. Результаты линейного алгоритма PCA показаны в левом столбце на рис. 8.14. Он имеет только два собственных значения, поскольку размерность входного пространства равна двум. В противоположность этому PCA на основе ядра позволяет извлекать компоненты более высокого порядка, как показано в столбцах 2–4 на рис. 8.14. Контурные линии, отображаемые в каждой из частей рисунка (за исключением нулевого собственного значения в линейном PCA), представляют собой постоянные главные значения (т.е. постоянные проекции на собственный вектор, связанный с рассматриваемым собственным значением).

На основе результатов, показанных на рис. 8.14, можно сделать следующие выводы.

Как и ожидалось, линейный PCA не способен обеспечить адекватное представление нелинейных входных данных.

В любом случае первый главный компонент монотонно изменяется вдоль параболы, окаймляющей входные данные.

В PCA на основе ядра второй и третий главные компоненты демонстрируют поведение, в чем-то аналогичное для различных степеней полинома d.

В случае степени полинома $d = 2$ третий главный компонент ядра PCA воспроизводит дисперсию, полученную вследствие аддитивного гауссова шума v. Удаляя из результата вклад этого компонента, получим эффект шумоподавления. ■

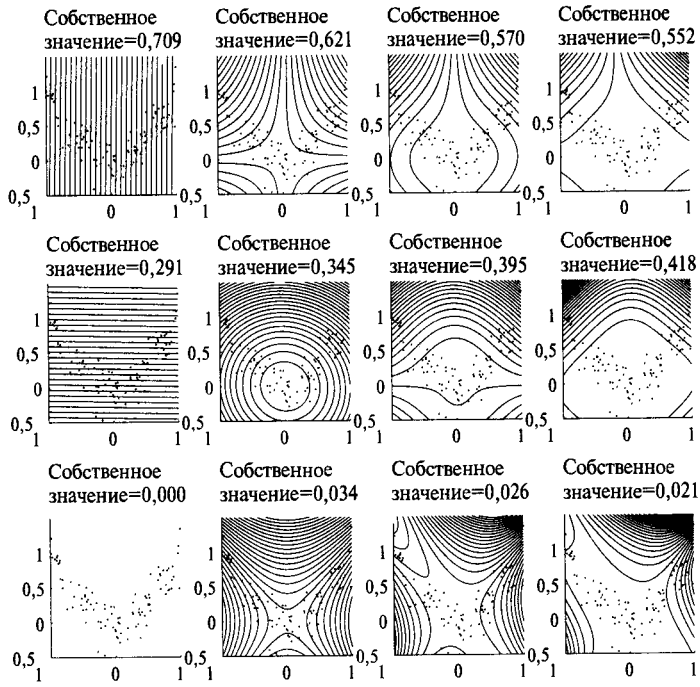


Рис. 8.14. Двумерный пример, иллюстрирующий работу алгоритма PCA на основе ядра. Слева направо показана степень полинома ядра $d = 1, 2, 3, 4$. Сверху вниз показаны три первых собственных вектора пространства признаков. Первый столбец соответствует обычному PCA, а остальные три — PCA на основе ядра со степенью полинома $d = 2, 3, 4$. *(Воспроизведено с разрешения доктора Клауса-Роберта Мюллера (Dr. Klaus-Robert Muller))*

8.11. Резюме и обсуждение

В этой главе представлен материал по теории анализа главных компонентов и показано использование нейросетей в реализации этого метода. Совершенно естественным сейчас будет следующий вопрос: "Насколько полезен анализ главных компонентов?" Ответ на этот вопрос зависит от конкретной области применения.

Если наша цель — добиться хорошего сжатия данных при одновременном сохранении как можно большей части информации о входном сигнале, то анализ главных компонентов обеспечит полезную самоорганизующуюся процедуру обучения. Следует подчеркнуть, что метод пространственной декомпозиции (см. раздел 8.3), основанный на "первых l главных компонентах" входных данных, осуществляет линейное отображение, являющееся оптимальным в том смысле, что позволяет реконструировать исходные входные данные в смысле среднеквадратической ошибки. Более того, представление, основанное на "первых l главных компонентах", является

предпочтительным для представления произвольного подпространства, так как главные компоненты входных данных являются естественным образом упорядоченными по убыванию собственных значений, или, что эквивалентно, по убыванию дисперсии. Следовательно, использование анализа главных компонентов для сжатия данных можно оптимизировать, взяв на вооружение наибольшую числовую точность для кодирования первого главного компонента и постепенно снижая ее для кодирования остальных $l - 1$ компонентов.

Связанным вопросом является представление множества данных, созданного из совокупности нескольких кластеров. Для того чтобы эти кластеры стали различимыми, расстояние между ними должно быть больше внутреннего разброса данных в каждом из кластеров. Если случилось так, что в множестве данных имеется несколько кластеров, то нужно найти основные оси. Для этого используется анализ главных компонентов, выбирающий проекции кластеров с хорошим разделением, создавая, таким образом, хороший базис для извлечения признаков.

В этом, последнем контексте упомянем одно полезное приложение анализа главных компонентов — *препроцессор* (preprocessor) для нейронной сети, обучаемой с учителем (например, многослойного персептрона, обучаемого по алгоритму обратного распространения). При этом целью является ускорение сходимости процесса обучения за счет декорреляции входных данных. Процедура обучения с учителем, какой является алгоритм обратного распространения, основана на принципе наискорейшего спуска. Процесс сходимости в этом виде обучения обычно замедляется из-за воздействия на синаптические веса многослойного персептрона посторонних шумов, даже при использовании таких процедур локального ускорения, как применение индивидуальных параметров скорости обучения и момента к отдельным весам. Однако если входной сигнал многослойного персептрона состоит из некоррелированных компонентов, то матрица Гессиана функции стоимости $\mathbf{E}(n)$ будет более близка к диагональной, чем в противоположном случае (см. главу 4). При этой форме диагонализации использование простых процедур локальной акселерации позволяет значительно ускорить процесс сходимости, что возможно с помощью независимого масштабирования параметров обучения вдоль каждой из осей весов [112].

Хеббовские алгоритмы обучения, описанные в этой главе, навеяны идеями нейробиологии, поэтому будет логичным завершить наш экскурс комментарием о роли анализа главных компонентов в биологических системах восприятия. В [648] был поставлен вопрос "существенности" анализа главных компонентов в качестве принципа определения свойств отклика, вырабатываемого нейроном в процессе анализа входных "сцен". В частности, оптимальность анализа главных компонентов по сравнению с точной реконструкцией входного сигнала из отклика нейрона является спорным вопросом. В общем случае оказывается, что мозг не просто пытается воспроизвести входные сцены, полученные от органов чувств, а выделяет некоторые "значимые намеки" или признаки, чтобы интерпретировать входные сигналы на более высоком

уровне. Таким образом, можно немного конкретизировать вопрос, поднятый в начале наших рассуждений, и спросить: "Насколько практичным является анализ главных компонентов в задаче обработки входных данных?"

В [41] указывается на значимость множества алгоритмов, предложенных в [798], [931], для анализа главных компонентов (т.е. для алгоритмов Хебба, рассмотренных в разделах 8.4 и 8.5), в *алгоритме иерархической кластеризации* (hierarchical clustering algorithm). Они продвигают гипотезу о том, что иерархическая кластеризация может выступать в качестве фундаментального свойства (по крайней мере, частично) памяти, которое можно использовать для распознавания свойств окружающей среды. Отсюда следует вывод, что самоорганизующийся анализ главных компонентов может играть существенную роль в иерархической кластеризации обучаемых областей церебральной коры мозга не из-за его свойств оптимальности восстановления, а благодаря его встроенному свойству создавать проекции кластеров с хорошим разделением.

Еще одно интересное приложение анализа главных компонентов к обработке входных данных заключается в решении задачи восстановления формы по тени, описанной в [80]. Эту задачу можно сформулировать так: "Каким образом мозг восстанавливает трехмерные формы на основе затененных образов на двухмерных изображениях?" В вышеуказанной работе для задачи восстановления формы предметов по затененным изображениям было предложено иерархическое решение, состоящее из двух этапов.

1. Посредством эволюции или накопленного опыта мозг обнаруживает, что объекты можно классифицировать на классы объектов меньшей размерности, принимая в расчет их формы.
2. На основании п. 1 извлечение формы из полутоновых образов сводится к более простой *задаче оценки параметров* в пространстве более низкой размерности.

Например, общая форма головы человека остается неизменной в том смысле, что все люди имеют выступ в виде носа, глазные впадины, а также лоб и щеки, представляющие собой ровные поверхности. Эти неизменные характеристики наводят на мысль, что любое лицо, выраженное функцией $r(\theta, l)$ в цилиндрических координатах, может быть представлено в виде суммы двух компонентов:

$$r(\theta, l) = r_0(\theta, l) + \rho(\theta, l),$$

где $r_0(\theta, l)$ — усредненная форма головы для данной категории людей (например, взрослых мужчин или взрослых женщин); $\rho(\theta, l)$ — отклонения, позволяющие идентифицировать конкретного человека. Обычно отклонение $\rho(\theta, l)$ является малым по отношению к $r_0(\theta, l)$. Для представления $\rho(\theta, l)$ в [80] использовался анализ главных компонентов, в котором отклонения описывались множеством собственных функций

(т.е. двухмерных составляющих собственных векторов). Результаты, представленные в [80], показали применимость двухшагового иерархического подхода для восстановления трехмерной поверхности лица человека по его двухмерному портрету.

Задачи

Фильтр Хебба для извлечения максимального собственного значения

8.1. В *согласованном фильтре* (matched filter) из примера 8.2 собственное значение λ_1 и соответствующий собственный вектор $\mathbf{q}_1$ определяются следующими формулами:

$$\lambda_1 = 1 + \sigma^2,$$
$$\mathbf{q}_1 = \mathbf{s}.$$

Покажите, что эти параметры удовлетворяют основному соотношению

$$\mathbf{R}\mathbf{q}_1 = \lambda_1 \mathbf{q}_1,$$

где $\mathbf{R}$ — матрица корреляции входного вектора $\mathbf{X}$.

8.2. Рассмотрим фильтр для извлечения максимального собственного значения, в котором вектор весов $\mathbf{w}(n)$ изменяется по формуле (8.46). Покажите, что дисперсия выхода этого фильтра достигает значения $\lambda_{\max}$ при n, равном бесконечности, где $\lambda_{\max}$ — наибольше собственное значение матрицы корреляции входного вектора.

8.3. Анализ второстепенных компонентов (MCA) противоположен анализу главных компонентов. Алгоритм MCA позволяет определить те направления, которые *минимизируют* дисперсию проекции. Эти направления являются собственными векторами, соответствующими наименьшим собственным значениям матрицы корреляции $\mathbf{R}$ входного вектора $\mathbf{X}(n)$.

В данной задаче требуется определить, каким образом следует модифицировать обособленный нейрон (см. раздел 8.4), чтобы найти второстепенные компоненты $\mathbf{R}$. В частности, можно изменить знак в правиле обучения (8.40), получив соотношение [1170]

$$w_i(n+1) = w_i(n) - \eta y(n)[x_i(n) - y(n)w_i(n)].$$

Покажите, что если наименьшим собственным значением матрицы $\mathbf{R}$ с кратностью 1 является λ_m, то

$$\lim_{n\to\infty} \mathbf{w}(n) = \eta \mathbf{q}_m,$$

где $\mathbf{q}_m$ — собственный вектор, соответствующий λ_m.

Анализ главных компонентов на основе правила Хебба

8.4. Постройте граф передачи сигнала, представляющий векторные уравнения (8.87) и (8.88).

8.5. Подход с использованием обычных дифференциальных уравнений при анализе сходимости (см. раздел 8.4) не применяется непосредственно к обобщенному алгоритму обучения Хебба. Однако, выразив матрицу синаптических весов $\mathbf{W}(n)$ (см. (8.91)) в виде вектора, составленного из ее столбцов, функцию коррекции $h(\cdot,\cdot)$ можно интерпретировать обычным образом, а затем применить теорему об асимптотической устойчивости. Основываясь на вышеизложенном, сформулируйте теорему о сходимости для обобщенного алгоритма обучения Хебба.

8.6. В этой задаче мы исследуем обобщенный алгоритм Хебба для случая двумерных рецепторных полей, генерируемых случайным входным сигналом [930]. Этот случайный вектор состоит из двумерного поля независимого гауссова шума с нулевым средним значением и единичной дисперсией, *свернутого* (convolved) гауссовой маской (фильтром) и умноженного на гауссово окно (Gaussian window). гауссова маска имеет стандартное отклонение в 2 пикселя, а гауссово окно — в 8 пикселей. Полученный в результате случайных вход $x(r,s)$ в точке (r,s) можно представить в следующем виде:

$$x(r,s) = m(r,s)[g(r,s) * w(r,s)],$$

где $w(r,s)$ — поле независимого гауссова шума; $g(r,s)$ — гауссова маска; $m(r,s)$ — функция гауссова окна. Функция *круговой свертки* (circular convolution) $g(r,s)$ и $w(r,s)$ определяется следующим образом:

$$g(r,s) * w(r,s) = \sum_{p=0}^{N-1}\sum_{q=0}^{N-1} g(p,q)w(r-p, s-q),$$

где предполагается, что $g(r,s)$ и $w(r,s)$ — периодические функции.

Используйте 2000 примеров случайного входного сигнала $x(r, s)$ для обучения однослойной сети прямого распространения с помощью обобщенного алгоритма Хебба. Данная сеть имеет 4096 входов, скомпонованных в ячейки пикселей размером 64×64, и 16 выходов. Полученные в результате обучения веса сети можно представить в виде матрицы размерности 64×64. Выполните описанные здесь вычисления и представьте 16 массивов синаптических весов в виде двумерных масок. Прокомментируйте результат.

8.7. Уравнение (8.113) представляет собой преобразованную версию уравнения коррекции (8.106) для вычисления вектора весов обратных связей $\mathbf{a}_j(n)$. Это преобразование основано на определении вектора синаптических весов $\mathbf{w}_j(n)$ в терминах m главных *мод* (modes) сети, представленных уравнением (8.109). Выведите уравнение (8.113).

8.8. Рассмотрите матрицу системы (8.116), представленную графом передачи сигнала на рис. 8.12, где $1 \leq k \leq j-1$.

а) Выведите характеристическое уравнение для этой матрицы размером 2×2.

б) Покажите, что эта матрица имеет двойное собственное значение.

в) Докажите утверждение, что основные моды сети имеют одно и то же собственное значение.

8.9. Алгоритм GHA использует только прямые связи, в то время как алгоритм APEX использует как прямые, так и латеральные связи. Несмотря на эти различия, сходимость алгоритма APEX теоретически является такой же, как и для алгоритма GHA. Докажите истинность этого утверждения.

PCA на основе ядра

8.10. Пусть $\bar{K}_{ij}$ — центрированная составляющая ij-го элемента матрицы ядра **K**. Покажите, что [945]

$$\bar{K}_{ij} = K_{ij} - \frac{1}{N}\sum_{m=1}^{N}\varphi^T(\mathbf{x}_m)\varphi(\mathbf{x}_j) - \frac{1}{N}\sum_{n=1}^{N}\varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}_n) + \\ + \frac{1}{N^2}\sum_{m=1}^{N}\sum_{n=1}^{N}\varphi^T(\mathbf{x}_m)\varphi(\mathbf{x}_n).$$

Запишите компактное представление этого соотношения в матричном виде.

8.11. Покажите, что нормализация собственного вектора $\boldsymbol{\alpha}$ матрицы ядра **K** эквивалентна удовлетворению требования (8.153).

8.12. Перечислите свойства PCA на основе ядра.

Карты самоорганизации

9.1. Введение

В этой главе мы продолжим изучение самоорганизующихся систем и рассмотрим особый класс искусственных нейронных сетей, который носит название карт самоорганизации. Эти сети основаны на *конкурентном обучении* (competitive learning). Отдельные нейроны выходного слоя такой сети соревнуются за право активации, в результате чего активным оказывается один нейрон в сети (или в группе). Выходной нейрон, который выиграл данное соревнование, называется *победившим* (winning neuron). Одним из способов организации такой конкуренции между нейронами является использование латеральных (т.е. отрицательных обратных) связей между ними. Эта идея была впервые предложена Розенблаттом [902].

В *картах самоорганизации* (self-organizing map) нейроны помещаются в узлах *решетки* (lattice), обычно одно- или двухмерной. Карты более высокой размерности также возможны, но используются достаточно редко. Нейроны в ходе конкурентного процесса избирательно настраиваются на различные входные образы (возбудители) или классы входных образов. Положения таким образом настроенных нейронов (т.е. нейронов-победителей) упорядочиваются по отношению друг к другу так, что на решетке создается значимая система координат [573]. Таким образом, самоорганизующиеся системы характеризуются формированием *топографических карт* (topographic map) входных образов, в которых *пространственное местоположение (т.е. координаты) нейронов решетки является индикатором встроенных статистических признаков, содержащихся во входных примерах*. Отсюда берет свое происхождение и само название "самоорганизующиеся карты".

В качестве нейронных моделей самоорганизующиеся карты реализуют мост между двумя уровнями адаптации.

- Правилами адаптации, сформулированными на микроуровне одного нейрона.
- Объединениями экспериментально лучших и физически доступных моделей извлечения признаков микроуровней слоев нейронов.

Так как карты самоорганизации по своей природе нелинейны, их можно рассматривать как нелинейное обобщение анализа главных компонентов [887].

Разработка карт самоорганизации в качестве нейронных моделей обусловлена следующим отличительным свойством человеческого мозга: он организован таким образом, что отдельные сенсорные входы представляются *топологически упорядоченными вычислительными картами* (topologically ordered computational map) в определенных его областях. В частности, такие сенсорные входы, как нервные окончания тактильной системы [530], зрения [489], [490] и слуха [1029], топологически упорядоченно отображаются на различные контуры церебральной коры мозга. Таким образом, вычислительное отображение является кирпичиком в инфраструктуре обработки информации нервной системой. Карта вычислений, в свою очередь, образована массивом нейронов, представляющих собой несколько по-разному настроенные процессоры или фильтры, параллельно принимающие информацию от различных сенсоров. Следовательно, нейроны преобразовывают входные сигналы в *пространственно-кодированные* (place-coded) *распределения вероятности*, представляющие вычисленные значения параметров узлами относительных максимумов активности [564]. К таким образом организованной информации получают доступ процессоры более высокого уровня, которые используют относительно простые схемы соединений.

Структура главы

Материал, представленный в настоящей главе, посвященной картам самоорганизации, расположен в следующем порядке. В разделе 9.2 описываются две модели отображения признаков, которые, каждая своим способом, способны извлекать существенные признаки вычислительных карт в биологической структуре мозга. Эти две модели отличаются друг от друга используемой формой представления входных сигналов.

Оставшаяся часть главы посвящена детальному изложению одной из этих моделей, получившей в работе Кохонена [579] название *самоорганизующейся карты*. В разделе 9.3 рассматриваются нейробиологические предположения, участвующие в математической формулировке модели Кохонена. Эта модель в сводном виде описывается в разделе 9.4. Особо важные свойства модели будут представлены в разделе 9.5, а в разделе 9.6 будет проведено компьютерное моделирование. Производительность карт признаков может быть настроена с помощью одного из приемов обучения с учителем, получившего название *квантования вектора обучения*. Этот прием описывается в разделе 9.7. В следующем разделе 9.8 будет проведен компьютерный эксперимент по адаптивной классификации моделей, который комбинирует в себе использование квантования векторов обучения и самоорганизующихся карт сжатия данных. В разделе 9.10 будет описано еще одно приложение карт самоорганизации – построение контекстных карт, которые находят применение в задачах фонетической классификации текста, наблюдения Земли и исследования данных. Как всегда, глава завершается разделом выводов и обсуждений (9.12).

9.2. Две основные модели отображения признаков

Любой исследователь человеческого мозга не может не поражаться доминирующей ролью его церебральной коры. Деятельность мозга практически целиком сосредоточена в коре, обходя другие возможные пути. Своей исключительной сложностью кора головного мозга превосходит, пожалуй, любую другую известную в природе структуру [489]. Равно поражает и *упорядоченность отображения* входов различных сенсоров (моторных, слуховых, зрительных и т.п.) в соответствующие области коры головного мозга. Хорошей иллюстрацией этого момента может служить приведенная на рис. 2.4 (см. главу 2) цитоархитектурная карта коры головного мозга. Вычислительные карты обладают следующими свойствами [564].

- На каждом этапе представления каждый поступающий фрагмент информации сохраняется в своем контексте.
- Нейроны, работающие с близко расположенными областями информации, также расположены достаточно близко друг к другу, таким образом взаимодействуя друг с другом посредством коротких синаптических связей.

Областью нашего интереса является построение искусственных топографических карт, которые обучаются посредством самоорганизации. Эта идея была навеяна исследованиями в области нейробиологии. В этом контексте, из приведенного краткого описания вычислительных карт, функционирующих в структуре мозга, вытекает одно очень важное заключение [573].

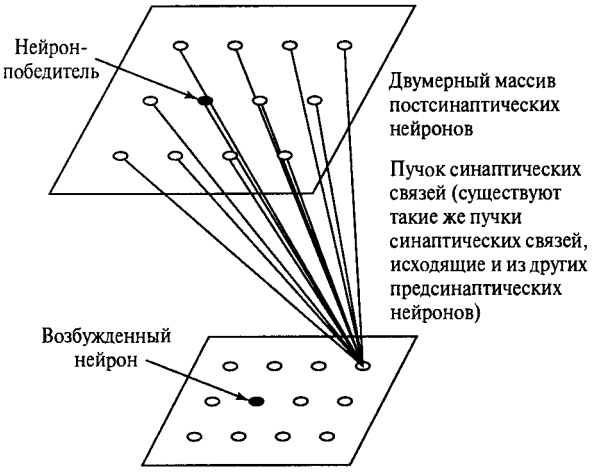
Пространственное положение выходных нейронов в топографической карте соответствует конкретной области признаков данных, выделенных из входного пространства.

В этом принципе сформулирован нейробиологический контекст, заложенный в двух совершенно различных *моделях отображения признаков* (feature-mapping model)[1], которые будут описаны ниже.

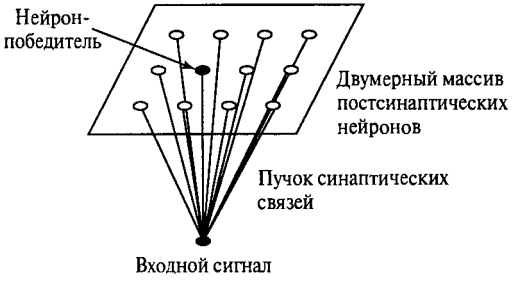
На рис. 9.1 показаны структурные схемы этих двух моделей. В обоих случаях выходные нейроны организованы в виде двумерной решетки. Такой тип топологии гарантирует, что каждый нейрон имеет множество соседей. Эти модели отличаются друг от друга способом задания входных образов.

Модель, показанная на рис. 9.1, *а*, была предложена в работе, посвященной биологическим основам, которые объясняли отображение сетчатки глаза на зрительную область коры головного мозга [1159]. В частности, существуют две связанные друг с другом двумерные решетки нейронов, где одна проецируется на другую. Первая ре-

[1]Две модели извлечения признаков, которые показаны на рис. 9.1, навеяны новаторскими исследованиями самоорганизации в [1100], где отмечалось, что модель зрительной области коры головного мозга (visual cortex



а) Модель Уилшоу-ван дер Мальсбурга



б) Модель Кохонена

Рис. 9.1. Две самоорганизующиеся карты признаков

шетка представлена предсинаптическими (входными), а вторая — постсинаптическими (выходными) нейронами. Постсинаптическая решетка использует *возбуждающий механизм близкого радиуса действия* (short-range exdcitatory mechanism), равно как и *тормозящий механизм дальнего действия* (long-range inhibitory mechanism). Эти два механизма по своей природе являются локальными и исключительно важными для самоорганизации. Решетки соединяются изменяемыми синапсами Хеббовского типа. Грубо говоря, постсинаптические нейроны не относятся к типу "победитель забирает все" — вместо этого используется порог, гарантирующий, что в любой момент времени будут активны только несколько постсинаптических нейронов. Более того, чтобы избежать неограниченного наращивания синаптических весов, которое может привести к неустойчивости системы, общий вес, ассоциированный с каждым постсинаптическим нейроном, ограничивается некоторым значением[2]. Таким образом,

[2] В [29] ограничения, накладываемые на синаптические веса постсинаптических нейронов, были несколько ослаблены. Математический анализ, представленный в этой работе, объясняет динамическую устойчивость карты коры мозга, сформированной в результате самоорганизации.

рост одного синаптического веса нейрона приводит к автоматическому уменьшению остальных. Основная идея *модели Уилшоу–ван дер Мальсбурга* состоит в том, что для представления информации о геометрической близости предсинаптических нейронов в форме их электрической активности и для использования этих корреляций в постсинаптической решетке соседние предсинаптические нейроны должны связываться с соседними постсинаптическими. Таким образом, самоорганизация приводит к топологически упорядоченному отображению. Однако следует заметить, что модель Уилшоу–ван дер Мальсбурга пригодна только для представления отображений, в которых размерности входного и выходного сигналов равны.

Вторая модель, показанная на рис. 9.1, *б*, была предложена Кохоненом [579]. Она не концентрируется на нейробиологических деталях. Эта модель извлекает существенные признаки вычислительных карт мозга и при этом остается вычислительно трактуемой[3]. Модель Кохонена по сравнению с моделью Уилшоу–ван дер Мальсбурга является более общей в том смысле, что она способна осуществлять сжатие данных (т.е. сокращение размерности входного сигнала).

Модель Кохонена принадлежит к классу *алгоритмов векторного кодирования* (vector-coding algorithm). Эта модель реализует топологическое отображение, которое оптимально размещает фиксированное количество векторов (т.е. кодовых слов) во входное пространство более высокой размерности и, таким образом, облегчает сжатие данных. Исходя из этого, модель Кохонена можно вывести двумя способами. Можно использовать базовые идеи самоорганизации, обусловленные нейробиологическими наблюдениями. Это — обычный подход [568], [573], [579]. А можно использовать подход векторного квантования, включающий кодирование и декодирование. Этот подход основывается на положениях теории коммуникаций [688], [691]. В этой главе мы рассмотрим оба подхода.

В литературе больше внимания уделяется модели Кохонена, чем модели Уилшоу–ван дер Мальсбурга, так как первая обладает некоторыми свойствами (о них мы поговорим немного позже в этой главе), которые имеют практический интерес для понимания и моделирования карт коры головного мозга. Вся оставшаяся часть настоящей главы будет посвящена изучению карт самоорганизации, а также их основных свойств и модификаций.

9.3. Карты самоорганизации

Основной целью карт самоорганизации (self-organizing map — SOM) является преобразование поступающих векторов сигналов, имеющих произвольную размерность, в одно- или двухмерную дискретную карту. При этом такое преобразование осуществ-

[3] Нейробиологические возможности самоорганизующейся карты (self-organizing map — SOM) обсуждаются в [568], [572].

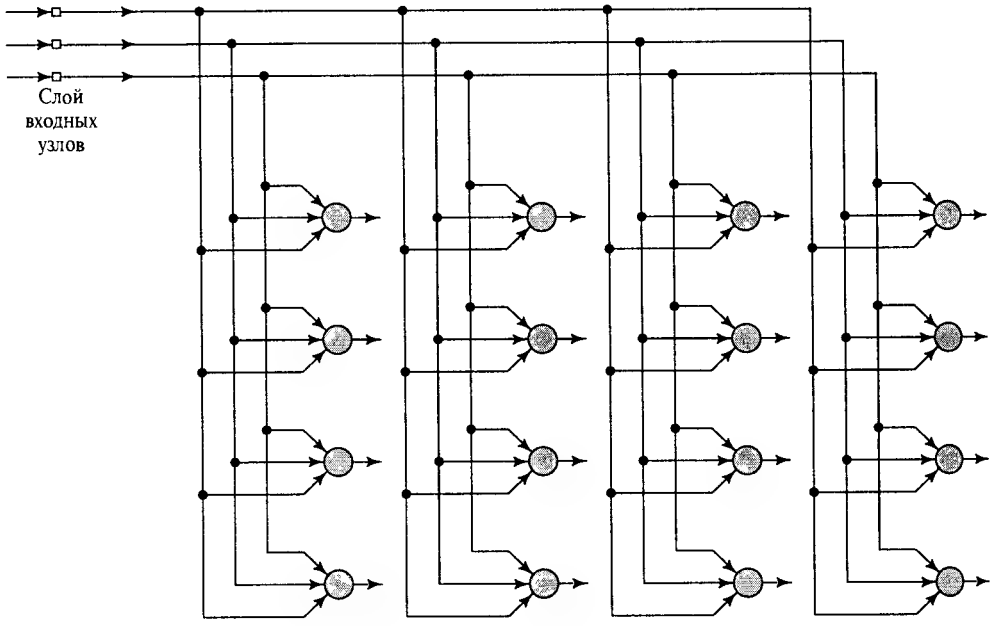


Рис. 9.2. Двумерная решетка нейронов

ляется адаптивно, в топологически упорядоченной форме. На рис. 9.2 показана схематическая диаграмма двумерной решетки нейронов, используемой в качестве дискретной карты. Все нейроны этой решетки связаны со всеми узлами входного слоя. Эта сеть имеет структуру прямого распространения с одним вычислительным слоем, состоящим из нейронов, упорядоченных в столбцы и строки. Одномерная решетка является частным случаем конфигурации, показанной на рис. 9.2. В этом частном случае вычислительный слой состоит всего из одной строки (или столбца) нейронов.

Каждый из входных слоев, представленных в сети, обычно состоит из локализованной области активности, размещаемой в относительно спокойной области. Расположение и природа такой области обычно варьируется в зависимости от конкретной реализации входного примера. Таким образом, для правильного развития процесса самоорганизации требуется, чтобы все нейроны сети были обеспечены достаточным количеством различных реализаций входных образов.

Алгоритм, ответственный за формирование самоорганизующихся карт, начинается с инициализации синаптических весов сети. Обычно это происходит с помощью назначения синаптическим весам малых значений, сформированных генератором случайных чисел. При таком формировании карта признаков изначально не имеет какого-либо порядка признаков. После корректной инициализации сети для формирования карты самоорганизации запускаются три следующих основных процесса.

1. *Конкуренция* (competition). Для каждого входного образа нейроны сети вычисляют относительные значения дискриминантной функции. Эта функция является основой конкуренции среди нейронов.
2. *Кооперация* (cooperation). Победивший нейрон определяет пространственное положение топологической окрестности нейронов, обеспечивая тем самым базис для кооперации между этими нейронами.
3. *Синаптическая адаптация* (synaptic adaptation). Последний механизм позволяет возбужденным нейронам увеличивать собственные значения дискриминантных функций по отношению к входным образам посредством соответствующих корректировок синаптических весов. Корректировки производятся таким образом, чтобы отклик нейрона-победителя на последующее применение аналогичных примеров усиливался.

Процессы конкуренции и кооперации осуществляются в соответствии с двумя из четырех принципов самоорганизации, описанных в главе 8. Что же касается принципа самоусиления, то он реализуется в модифицированной форме — форме обучения Хебба для адаптивного процесса. Как было отмечено в главе 8, для обучения необходима избыточность входных данных (хотя на этом при описании алгоритма SOM особо не заостряется внимание), так как это реализует процесс извлечения знаний. Детальные описания процессов конкуренции, кооперации и синаптической адаптации будут представлены ниже.

Процесс конкуренции

Пусть m — размерность входного пространства, а входной вектор выбирается из этого входного пространства случайно и обозначается так:

$$\mathbf{x} = [x_1, x_2, \ldots, x_m]^T. \tag{9.1}$$

Вектор синаптических весов каждого из нейронов сети имеет ту же размерность, что и входное пространство. Обозначим синаптический вес нейрона j следующим образом:

$$\mathbf{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T, j = 1, 2, .., l, \tag{9.2}$$

где l — общее количество нейронов сети. Для того чтобы подобрать наилучший вектор $\mathbf{w}_j$, соответствующий входному вектору $\mathbf{x}$, нужно сравнить скалярные произведения $\mathbf{w}_j^T\mathbf{x}$ для $j = 1, 2, \ldots, l$ и выбрать наибольшее значение. При этом предполагается, что ко всем нейронам применяется некоторое значение *насыщения* (threshold). Эта величина равна *порогу* (bias), взятому с обратным знаком. Таким образом, выбирая нейрон с наибольшим скалярным произведением $\mathbf{w}_j^T\mathbf{x}$, мы в результате определяем

местоположение, которое должно стать центром топологической окрестности возбужденного нейрона.

Как говорилось в главе 1, наилучший критерий соответствия, основанный на максимизации скалярного произведения $\mathbf{w}_j^T\mathbf{x}$, математически эквивалентен минимизации Евклидова расстояния между векторами $\mathbf{x}$ и $\mathbf{w}_j$. Если использовать индекс $i(\mathbf{x})$ для идентификации того нейрона, который лучше всего соответствует входному сигналу $\mathbf{x}$, то эту величину можно определить с помощью следующего соотношения[4]:

$$i(\mathbf{x}) = \arg\min_j \|\mathbf{x} - \mathbf{w}_j\|, \quad j = 1, 2, \ldots, l. \tag{9.3}$$

В этом выражении проявляется сущность всего процесса конкуренции между нейронами. Согласно выражению (9.3), предметом внимания является величина $i(\mathbf{x})$. Конкретный нейрон i, удовлетворяющий данному условию, называется *победившим* (winning) или *наиболее подходящим* (best-matching) для данного входного вектора $\mathbf{x}$. Уравнение (9.3) становится источником следующего наблюдения.

Непрерывное входное пространство образов активации нейронов с помощью процесса конкуренции между отдельными нейронами сети отображается в дискретное выходное пространство.

В зависимости от приложения данного метода откликом сети может быть либо индекс победившего нейрона (т.е. его позиция в решетке), либо вектор синаптических весов, являющийся самым близким к входному в Евклидовом смысле.

Процесс кооперации

Нейрон-победитель находится в центре топологической окрестности сотрудничающих нейронов. Возникает ключевой вопрос: как определить топологическую окрестность, которая будет корректна с нейробиологической точки зрения? Для того чтобы ответить на него, вспомним о нейробиологическом обосновании *латерального взаимодействия* (lateral interaction) среди множества возбужденных нейронов. В частности, возбужденный нейрон всегда пытается возбудить пространственно *близкие* к нему нейроны, и это интуитивно понятно. Это наблюдение приводит к определению топологической окрестности победившего нейрона j, которая плавно сходит на нет с увеличением расстояния[5] [668], [669], [891]. Для большей конкретизации обозначим

---

[4] Правило конкурентного (competitive) обучения (9.3) в литературе по нейронным сетям впервые было введено в работе Гроссберга [400].

[5] В своей исходной форме алгоритма SOM, предложенной в работе Кохонена [579], предполагалось, что топологическая окрестность имеет постоянную амплитуду. Пусть $d_{j,i}$ — *латеральное (lateral) расстояние* между победившим (j) и возбужденным (i) нейронами в функции окрестности. В таком случае топологическая окрестность в одномерной решетке определяется следующим образом:

$$h_{j,i} = \begin{cases} 1, & -K \le d_{j,i} \le K \\ 0, & \text{в противном случае,} \end{cases} \tag{1}$$

символом $h_{j,i}$ *топологическую окрестность* (topological neighbourhood) с центром в победившем нейроне i, состоящую из множества возбуждаемых (кооперирующихся) нейронов, типичный представитель которой имеет индекс j. Пусть $d_{j,i}$ — *латеральное расстояние* (lateral distance) между победившим (i) и вторично возбужденным (j) нейронами. Тогда можно предположить, что топологическая окрестность $h_{j,i}$ является унимодальной функцией латерального расстояния $d_{j,i}$ и удовлетворяет двум следующим требованиям.

- Топологическая окрестность $h_{j,i}$ является симметричной относительно точки максимума, определяемой при $d_{j,i} = 0$. Другими словами, максимум функции достигается в победившем нейроне.
- Амплитуда топологической окрестности $h_{j,i}$ монотонно уменьшается с увеличением латерального расстояния $d_{j,i}$, достигая нуля при $d_{j,i} \to \infty$. Это необходимое условие сходимости.

Типичным примером $h_{j,i}$, удовлетворяющим этим требованиям, является *функция Гаусса*[6]:

$$h_{j,i(\mathbf{x})} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right). \tag{9.4}$$

Это выражение является инвариантным к расположению победившего нейрона. Параметр σ называют *эффективной шириной* (effective width) топологической окрестности (рис. 9.3). Этот параметр определяет уровень, до которого нейроны из топологической окрестности победившего участвуют в процессе обучения. В качественном смысле гауссова топологическая окрестность (согласно (9.4)) является более подходящей с биологической точки зрения, чем прямоугольная.

Она также приводит к более быстрой сходимости алгоритма SOM, чем прямоугольная топологическая окрестность [284], [668], [669].

Для обеспечения кооперации между соседними нейронами необходимо, чтобы топологическая окрестность $h_{j,i}$ была зависимой от латерального расстояния между победившим (i) и возбуждаемым (j) нейроном в выходном пространстве, а не от какой-либо меры длины в исходном входном пространстве. Именно это свойство отражено в выражении (9.4). В случае одномерной решетки расстояние $d_{j,i}$ является

где $2K$ — общий размер одномерной окрестности возбужденных нейронов. В противовес соглашениям из нейробиологии, сущность модели (1) состоит в том, что все нейроны, расположенные в топологической окрестности, возбуждаются на одном и том же уровне, а взаимодействие между ними не зависит от латерального расстояния до победившего нейрона i.

[6] В [285] было показано, что метастабильные состояния, представляющие топологические дефекты в конфигурации карты признаков, возникают в тех случаях, когда алгоритм SOM использует невыпуклую функцию окрестности. Функция Гаусса является выпуклой, в то время как прямоугольная область — нет. Широкая, выпуклая функция окрестности (такая как гауссова функция большого радиуса) приводит к более быстрому топологическому упорядочиванию, чем невыпуклая, так как она не содержит метастабильных состояний.

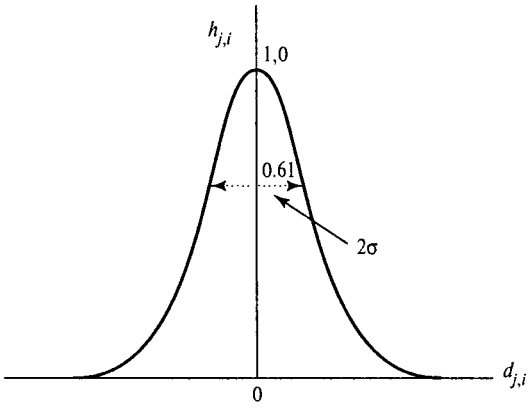


Рис. 9.3. Гауссова функция окрестности

целым числом, равным модулю $|j-i|$. В случае же двумерной решетки это расстояние определяется соотношением

$$d_{j,i}^2 = \|\mathbf{r}_j - \mathbf{r}_i\|^2, \tag{9.5}$$

где дискретный вектор $\mathbf{r}_j$ определяет позицию возбуждаемого нейрона j, а $\mathbf{r}_i$ – победившего нейрона i. Оба этих измерения проводятся в дискретном выходном пространстве.

Еще одним уникальным свойством алгоритма SOM является то, что размер топологической окрестности со временем уменьшается. Это требование удовлетворяется за счет постепенного уменьшения ширины σ функции топологической окрестности $h_{j,i}$. Популярным вариантом зависимости величины σ от дискретного времени n является экспоненциальное убывание, описываемое следующей формулой [792], [891]:

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right), \quad n = 0, 1, 2, \ldots, \tag{9.6}$$

где σ_0 — начальное значение величины σ в алгоритме SOM; τ_1 — некоторая временная константа. Исходя из этого, топологическая окрестность имеет форму, зависящую от времени, т.е.

$$h_{j,i(\mathbf{x})}(n) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(n)}\right), \quad n = 0, 1, 2 \ldots, \tag{9.7}$$

где $\sigma(n)$ определяется по формуле (9.6). Таким образом, при увеличении количества итераций n ширина $\sigma(n)$ экспоненциально убывает, а вместе с ней соответственно сжимается и топологическая окрестность. Далее функцию $h_{j,i(x)}(n)$ будем называть *функцией окрестности* (neighbourhood function).

Еще один неплохой способ рассмотрения динамики функции окрестности $h_{j,i(x)}(n)$ вокруг победившего нейрона $i(\mathbf{x})$ был предложен в [690]. В ней назначение функции $h_{j,i(x)}(n)$ сводилось к корреляции направлений векторов весов большого количества возбужденных нейронов в решетке. По мере уменьшения ширины $h_{j,i(x)}(n)$ направления коррекции нейронов коррелировались. Это явление становилось особенно очевидным, когда карта самоорганизации отображалась на графическом экране монитора. С учетом ограниченных ресурсов компьютера было бы довольно расточительным коррелированно изменять большое количество степеней свободы вокруг победившего нейрона, как происходит в обычном алгоритме SOM. Вместо этого лучше использовать *ренормализованную* (renormalized) форму обучения SOM, в соответствии с которой приходится работать с гораздо меньшим количеством *нормализованных степеней свободы* (normalized degree of freedom). Эту операцию легко выполнять в дискретной форме для функции окрестности $h_{j,i(x)}(n)$ с *постоянной* шириной, постепенно увеличивая общее количество нейронов. Новые нейроны вставляются между старыми, а свойства гладкости алгоритма SOM гарантируют, что эти новые нейроны грациозно вольются в процесс синаптической адаптации [690]. Ренормализованный алгоритм SOM в сжатом виде будет описан в задаче 9.13.

Процесс адаптации

Теперь приступим к последнему процессу самоорганизующегося формирования карты признаков — процессу синаптической адаптации. Для того чтобы сеть могла самоорганизоваться, вектор синаптических весов $\mathbf{w}_j$ нейрона j должен изменяться в соответствии с входным вектором $\mathbf{x}$. Вопрос сводится к тому, *каким* должно быть это изменение. В постулате обучения Хебба синаптический вес усиливался при одновременном возникновении предсинаптической и постсинаптической активности. Это правило хорошо подходит к ассоциативному обучению. Однако для типа обучения без учителя, которое мы рассматриваем в этой главе, гипотеза Хебба в своей основной форме не может удовлетворить требованиям задачи. На это есть своя причина: изменения в связях происходят только в одном направлении, что в конечном счете приводит все веса к точке насыщения. Для того чтобы обойти эту проблему, немного изменим гипотезу Хебба и введем в нее *слагаемое забывания* (forgetting term) $g(y_j)\mathbf{w}_j$, где $\mathbf{w}_j$ — вектор синаптических весов нейрона j; $g(y_j)$ — некоторая положительная скалярная функция отклика y_j. Единственным требованием, налагаемым на функцию $g(y_j)$, является равенство нулю постоянного слагаемого разложения в ряд Тейлора функции $g(y_j)$, что, в свою очередь, влечет выполнение соотношения

$$g(y_j) = 0 \text{ при } y_j = 0. \tag{9.8}$$

Значимость этого требования сразу же становится очевидной. Имея такую функцию, изменение вектора весов нейрона j в решетке можно выразить следующим образом:

$$\Delta \mathbf{w}_j = \eta y_j \mathbf{x} - g(y_j)\mathbf{w}_j, \tag{9.9}$$

где η — *параметр скорости обучения* (learning-rate parameter) алгоритма. Первое слагаемое в правой части (9.9) является слагаемым Хебба, а второе — слагаемым забывания. Для того чтобы удовлетворялось соотношение (9.8), выберем следующую линейную функцию $g(y_j)$:

$$g(y_j) = \eta y_j. \tag{9.10}$$

Выражение (9.9) можно упростить, приняв

$$y_j = h_{j,i(x)}. \tag{9.11}$$

Подставляя (9.10) и (9.11) в (9.9), получим:

$$\Delta \mathbf{w}_j = \eta h_{j,i(x)}(\mathbf{x} - \mathbf{w}_j). \tag{9.12}$$

В заключение, учитывая формализацию дискретного времени, для данного вектора синаптических весов $\mathbf{w}_j(n)$ в момент времени n обновленный вектор $\mathbf{w}_j(n+1)$ в момент времени $n+1$ можно определить в следующем виде [568], [579], [891]:

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n) h_{j,i(x)}(n)(\mathbf{x} - \mathbf{w}_j(n)). \tag{9.13}$$

Это выражение применяется ко всем нейронам решетки, которые лежат в топологической окрестности победившего нейрона i. Выражение (9.13) имеет эффект перемещения вектора синаптических весов $\mathbf{w}_i$ победившего нейрона i в сторону входного вектора $\mathbf{x}$. При периодическом представлении данных обучения благодаря коррекции в окрестности победившего нейрона векторы синаптических весов будут стремиться следовать распределению входных векторов. Таким образом, представленный алгоритм ведет к *топологическому упорядочиванию* (topological ordering) пространства признаков во входном пространстве, в том смысле, что нейроны, корректируемые в решетке, будут стремиться иметь одинаковые синаптические веса. В разделе 9.5 этот вопрос будет затронут несколько глубже.

Равенство (9.13) является формулой вычисления синаптических весов карты признаков. Однако в дополнение к этому уравнению для выбора функции окрестности $h_{j,i(x)}(n)$ требуется учитывать эвристику (9.7). Еще одна эвристика необходима для выбора параметра скорости обучения $\eta(n)$.

Параметр скорости обучения $\eta(n)$ согласно (9.13) должен зависеть от времени, как это и должно быть при стохастической аппроксимации. К примеру, можно начать с некоторого исходного значения η_0, а затем, с течением времени, постепенно уменьшать его. Это требование можно удовлетворить, выбрав для параметра интенсивности обучения следующую экспоненциальную функцию:

$$\eta(n) = \eta_0 \exp\left(-\frac{n}{\tau_2}\right), \quad n = 0, 1, 2, \ldots, \tag{9.14}$$

где τ_2 — еще одна временная константа алгоритма SOM. Формулы экспоненциального убывания параметров скорости обучения (9.14) и ширины функции окрестности (9.6) могут не являться оптимальными. Они просто адекватны процессу самоорганизующегося формирования карты признаков.

Два этапа адаптивного процесса: упорядочивание и сходимость

Учитывая исходное состояние полного беспорядка, довольно удивительным выглядит то, что алгоритм SOM постепенно достигает организованного представления моделей активации, взятых из входного пространства, при условии правильного выбора параметров этого алгоритма. Можно выполнить декомпозицию процесса адаптации синаптических весов сети, вычисляемых в соответствии с формулой (9.13), разбив его на эва этапа: этап самоорганизации, за которым следует этап сходимости. Эти два этапа адаптивного процесса можно описать следующим образом [568], [579].

1. Этап *самоорганизации* или *упорядочивания*. Во время первого этапа адаптивного процесса происходит топологическое упорядочивание векторов весов. В алгоритме SOM этот этап может занять до 1000 итераций, а возможно, и больше. При этом значительное внимание следует уделить выбору параметра скорости обучения и функции окрестности.
   - Параметр скорости обучения $\eta(n)$ лучше выбрать близким к значению 0,1. С течением времени он должен убывать, но оставаться больше величины 0,01. Этого можно добиться, применив в формуле (9.14) следующие значения констант:
   $$\eta_0 = 0,1,$$
   $$\tau_2 = 1000.$$
   - Функция окрестности $h_{j,i}(n)$ должна изначально охватывать практически все нейроны сети и иметь центр в победившем нейроне i. Со временем эта функция будет постепенно сужаться. В частности, на этапе упорядочивания, который может занять около 1000 итераций, $h_{j,i}(n)$, скорее всего, сократится до малого значения и будет содержать в себе только ближайших соседей победившего

нейрона, а возможно, только его самого. Для двумерной решетки нейронов можно установить исходное значение σ_0 функции окрестности, равное "радиусу" решетки. Соответственно константу времени τ_1 в формуле (9.6) можно определить следующим образом:

$$\tau_1 = \frac{1000}{\log \sigma_0}.$$

2. Этап *сходимости*. Второй этап адаптивного процесса требуется для точной подстройки карты признаков и, таким образом, — для точного квантования входного пространства. Как правило, количество итераций, достаточное для этапа сходимости, примерно в 500 раз превышает количество нейронов сети. Таким образом, количество итераций этапа сходимости может достигать тысяч и десятков тысяч.
   - Для хорошей статистической точности параметр скорости обучения $\eta(n)$ на время этого этапа должен быть установлен в достаточно малое значение (порядка 0,01). В любом случае его нельзя очень приближать к нулю, иначе сеть может заступориться в *метаустойчивом* (metastable) состоянии. Метаустойчивое состояние соответствует конфигурации карты признаков с топологическим дефектом. Экспоненциальное сокращение (9.14) гарантирует невозможность достижения метаустойчивых состояний.
   - Функция окрестности $h_{j,i}(n)$ должна охватывать только ближайших соседей победившего нейрона, количество которых может сократиться до одного или даже до нуля.

9.4. Краткое описание алгоритма SOM

Сущность алгоритма SOM, предложенного Кохоненом, состоит в простом геометрическом вычислении свойств Хеббоподобного правила обучения и латеральных взаимодействий. Существенными характеристиками этого алгоритма являются следующие.

- Непрерывное входное пространство образов активации, которые генерируются в соответствии с некоторым распределением вероятности.
- Топология сети в форме решетки, состоящей из нейронов. Она определяет дискретное выходное пространство.
- Зависящая от времени функция окрестности $h_{j,i(x)}(n)$, которая определена в окрестности нейрона-победителя $i(\mathbf{x})$.
- Параметр скорости обучения $\eta(n)$, для которого задается начальное значение η_0 и который постепенно убывает во времени n, но никогда не достигает нуля.

Для функции окрестности и параметра скорости обучения на этапе упорядочивания (т.е. приблизительно для первой тысячи итераций) можно использовать формулы (9.7) и (9.14) соответственно. Для хорошей статистической точности на этапе сходимости параметр $\eta(n)$ должен быть установлен в очень малое значение (0,01 или меньше). Что же касается функции окрестности, то в начале этапа сходимости она должна содержать только ближайших соседей нейрона-победителя (при этом может включать только его самого).

Не считая инициализации, алгоритм проходит три основных шага: *подвыборка* (sampling), *поиск максимального соответствия* (similarity matching) и *корректировка* (updating). Кратко весь алгоритм можно описать следующим образом.

1. *Инициализация.* Для исходных векторов синаптических весов $\mathbf{w}_j(0)$ выбираются случайные значения. Единственным требованием здесь является различие векторов для разных значений $j = 1, 2, \ldots, l$, где l — общее количество нейронов в решетке. При этом рекомендуется сохранять малой амплитуду значений.

   Еще одним способом инициализации алгоритма является случайный выбор множества весов $\{\mathbf{w}_j(0)\}_{j=1}^{l}$ из доступного множества входных векторов $\{\mathbf{x}_i\}_{i=1}^{N}$.
2. *Подвыборка.* Выбираем вектор $\mathbf{x}$ из входного пространства с определенной вероятностью. Этот вектор представляет собой возбуждение, которое применяется к решетке нейронов. Размерность вектора $\mathbf{x}$ равна m.
3. *Поиск максимального подобия.* Находим наиболее подходящий (победивший) нейрон $i(\mathbf{x})$ на шаге n, используя критерий минимума Евклидова расстояния:

$$i(\mathbf{x}) = \arg\min_j \|\mathbf{x} - \mathbf{w}_j\|, \quad j = 1, 2, \ldots, l.$$

4. *Коррекция.* Корректируем векторы синаптических весов всех нейронов, используя следующую формулу:

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n) h_{j,i(x)}(n)(\mathbf{x} - \mathbf{w}_j(n)),$$

   где $\eta(n)$ — параметр скорости обучения; $h_{j,i(x)}(n)$ — функция окрестности с центром в победившем нейроне $i(\mathbf{x})$. Оба этих параметра динамически изменяются во время обучения с целью получения лучшего результата.
5. *Продолжение.* Возвращаемся к шагу 2 и продолжаем вычисления до тех пор, пока в карте признаков не перестанут происходить заметные изменения.

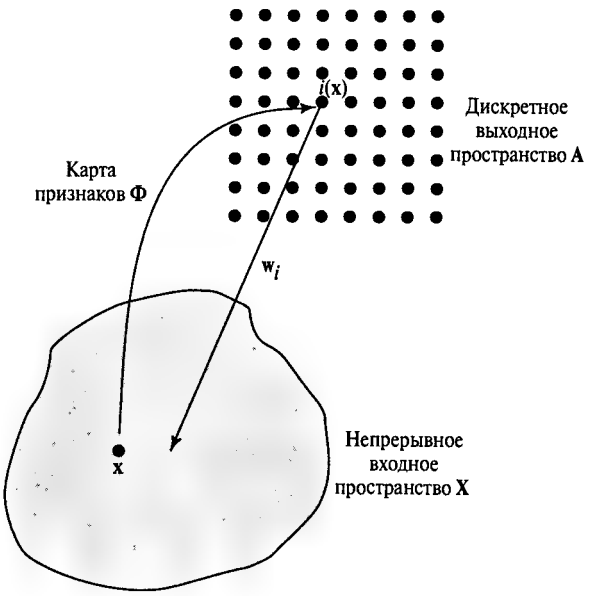


Рис. 9.4. Взаимосвязь между картой признаков Φ и вектором весов $\mathbf{w}_i$ победившего нейрона i

9.5. Свойства карты признаков

По завершении процесса сходимости алгоритма SOM вычисленная им *карта признаков* (feature map) отображает важные статистические характеристики входного пространства.

Для начала обозначим *пространственно непрерывное входное пространство (данных)* (spatially continous (data) space) символом $\mathbf{X}$. Его топология определяется метрическими связями векторов $\mathbf{x} \in \mathbf{X}$. Теперь обозначим символом $\mathbf{A}$ *пространственно дискретное выходное пространство* (spatially discrete output space). Его топология определяется упорядоченным множеством нейронов вычислительных узлов решетки. Обозначим символом Φ нелинейное преобразование (называемое *картой признаков*), которое отображает входное пространство $\mathbf{X}$ в выходное пространство $\mathbf{A}$:

$$\Phi : \mathbf{X} \rightarrow \mathbf{A}. \tag{9.15}$$

Выражение (9.15) можно рассматривать как соотношение (9.3) (определяющее положение нейрона $i(\mathbf{x})$, победившего при подаче на вход сети вектора $\mathbf{x}$) в абстрактной форме. Например, в контексте нейробиологии входное пространство $\mathbf{X}$ может представлять множество координат сомато-сенсорных рецепторов, распределенных по всей поверхности тела. Соответственно выходное пространство $\mathbf{A}$ представляет множество нейронов, расположенных в тех слоях коры головного мозга, которые отвечают за сомато-сенсорные рецепторы.

Рис. 9.5. Модель кодера-декодера

Для данного входного вектора $\mathbf{x}$ алгоритм SOM определяет наиболее подходящий (т.е. победивший) нейрон $i(\mathbf{x})$ в выходном пространстве $\mathbf{A}$, руководствуясь картой признаков $\mathbf{\Phi}$. Вектор синаптических весов $\mathbf{w}_i$ нейрона $i(\mathbf{x})$ после этого можно рассматривать как *указатель* на этот нейрон из входного пространства $\mathbf{X}$. Это значит, что синаптические элементы вектора $\mathbf{w}_i$ можно рассматривать как координаты *образа* нейрона i, проектируемые во входное пространство. Эти две операции показаны на рис. 9.4.

Карта признаков $\mathbf{\Phi}$ обладает следующими важными свойствами.

Свойство 1. Аппроксимация входного пространства. *Карта признаков* $\mathbf{\Phi}$, *представленная множеством векторов синаптических весов* $\{\mathbf{w}_j\}$, *в выходном пространстве* $\mathbf{A}$ *реализует хорошую аппроксимацию входного пространства* $\mathbf{X}$.

Главной целью алгоритма SOM является хранение большого объема векторов $\mathbf{x} \in \mathbf{X}$ с помощью нахождения небольшого набора прототипов $\mathbf{w}_j \in \mathbf{A}$, которые представляют собой адекватную аппроксимацию исходного входного пространства $\mathbf{X}$. Теоретические основы этой идеи уходят корнями в *теорию векторного квантования* (vector quantization theory). Ее мотивацией является снижение размерности или сжатие данных [346]. Кажется целесообразным представить в этой книге краткий экскурс в эту теорию.

Рассмотрим рис. 9.5, на котором $\mathbf{c}(\mathbf{x})$ выступает в роли *кодера* входного вектора $\mathbf{x}$, а $\mathbf{x}'(\mathbf{c})$ — *декодера* сигнала $\mathbf{c}(\mathbf{x})$. Вектор $\mathbf{x}$ случайным образом выбирается из множества примеров обучения (т.е. из входного пространства $\mathbf{X}$) согласно некоторой функции плотности вероятности $f_X(\mathbf{x})$. Оптимальная схема кодирования-декодирования определяется с помощью изменения функций $\mathbf{c}(\mathbf{x})$ и $\mathbf{x}'(\mathbf{c})$ с целью минимизации *ожидаемого искажения* (expected distortion), определяемого как

$$D = \frac{1}{2} \int_{-\infty}^{+\infty} d\mathbf{x} f_{\mathbf{X}}(\mathbf{x}) d(\mathbf{x}, \mathbf{x}'), \tag{9.16}$$

где множитель 1/2 введен для удобства выкладок, а $d(\mathbf{x}, \mathbf{x}')$ — *мера искажения* (distortion measure).

Интегрирование осуществляется по всему входному пространству $\mathbf{X}$, причем предполагается, что оно имеет размерность m. Популярным вариантом меры искажения

$d(\mathbf{x}, \mathbf{x}')$ является квадрат Евклидова расстояния между входным вектором $\mathbf{x}$ и восстановленным вектором $\mathbf{x}'$, т.е.

$$d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = (\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}'). \tag{9.17}$$

Исходя из этого, выражение (9.16) можно переписать в следующем виде:

$$D = \frac{1}{2}\int_{-\infty}^{+\infty} d\mathbf{x} f_{\mathbf{X}}(\mathbf{x}) \|\mathbf{x} - \mathbf{x}'\|^2. \tag{9.18}$$

Необходимые условия минимизации ожидаемого искажения D содержатся в *обобщенном алгоритме Ллойда* (generalized Lloyd algorithm)[7] [346]. Эти условия имеют следующий вид.

Условие 1. Для данного входного вектора $\mathbf{x}$ выбрать кодер $\mathbf{c} = \mathbf{c}(\mathbf{x})$, минимизирующий квадратичную ошибку искажения $||\mathbf{x} - \mathbf{x}'(\mathbf{c})||^2$.

Условие 2. Для данного кодера $\mathbf{c}$ вычислить восстановленный вектор $\mathbf{x}' = \mathbf{x}'(\mathbf{c})$ как центр тяжести (centroid) тех входных векторов $\mathbf{x}$, которые удовлетворяют условию 1.

Условие 1 называют *правилом ближайшего соседа* (nearest neighbour rule). Условия 1 и 2 подразумевают, что среднее искажение D стационарно (т.е. находится в точке локального минимума) по отношению к вариациям кодера $\mathbf{c}(\mathbf{x})$ и декодера $\mathbf{x}'(\mathbf{c})$ соответственно. Для того чтобы выполнить векторное квантование, обобщенный алгоритм Ллойда должен работать в *пакетном* режиме обучения. В своей основе алгоритм состоит из последовательной оптимизации кодера $\mathbf{c}(\mathbf{x})$ в соответствии с условием 1 и последующей оптимизации декодера $\mathbf{x}'(\mathbf{c})$ в соответствии с условием 2. Процесс продолжается до тех пор, пока ожидаемое искажение D не достигнет минимума. Для того чтобы избежать проблемы локального минимума, целесообразно запускать обобщенный алгоритм Ллойда несколько раз с различными начальными векторами.

Обобщенный алгоритм Ллойда тесно связан с алгоритмом SOM [691]. Форму этой связи можно описать, рассмотрев схему, показанную на рис. 9.6, где был введен независимый от сигнала *процесс шума* $\mathbf{v}$, который следует за кодером $\mathbf{c}(\mathbf{x})$. Шум $\mathbf{v}$ ассоциируется с фиктивным "каналом связи" между кодером и декодером. Это моделирует возможность того, что выходной код $\mathbf{c}(\mathbf{x})$ может быть искажен.

На основании модели, представленной на рис. 9.6, можно вывести *модифицированную* форму ожидаемого искажения в следующем виде:

$$D_1 = \frac{1}{2}\int_{-\infty}^{+\infty} d\mathbf{x} f_{\mathbf{X}}(\mathbf{x}) \int_{-\infty}^{\infty} d\mathbf{v}\pi(\mathbf{v}) \|\mathbf{x} - \mathbf{x}'(\mathbf{c}(\mathbf{x}) + \mathbf{v})\|^2, \tag{9.19}$$

---

[7] В литературе по теориям коммуникаций и информации для скалярного квантования предлагается более ранний метод, известный как *алгоритм Ллойда*. Этот алгоритм впервые был описан в 1957 году Ллойдом в неопубликованном отчете лаборатории Bell, а затем, намного позже, появился в печатном виде в [667]. Алгоритм Ллойда иногда еще называют алгоритмом квантования максимума

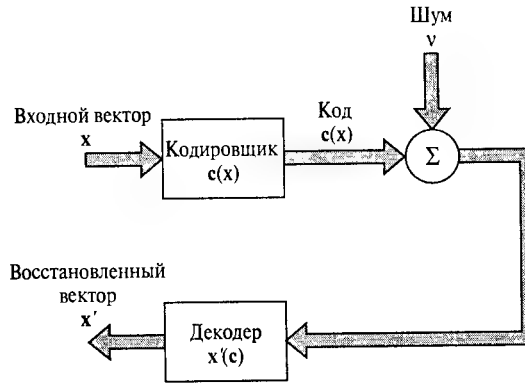


Рис. 9.6. Зашумленная модель кодера-декодера

где $\pi(\mathbf{v})$ — функция плотности вероятности аддитивного шума $\mathbf{v}$, а второе интегрирование выполняется по всем возможным реализациям этого шума.

В соответствии со стратегией, описанной для обобщенного алгоритма Ллойда, в модели, представленной на рис. 9.6, рассматриваются две различные задачи оптимизации: одна для кодера, а другая — для декодера. Чтобы найти оптимальный кодер для заданного вектора $\mathbf{x}$, нужно взять частную производную меры ожидаемого искажения D_1 по кодированному вектору $\mathbf{c}$. Используя (9.19), получим:

$$\frac{\partial D_1}{\partial \mathbf{c}} = \frac{1}{2} f_{\mathbf{X}}(\mathbf{x}) \int_{-\infty}^{+\infty} d\mathbf{v}\pi(\mathbf{v}) \frac{\partial}{\partial \mathbf{c}} \left\| \mathbf{x} - \mathbf{x}'(\mathbf{c}) \right\|^2 \Big|_{\mathbf{c}=\mathbf{c}(\mathbf{x})+\mathbf{v}}. \tag{9.20}$$

Чтобы найти оптимальный декодер для данного $\mathbf{c}$, нужно вычислить частную производную меры ожидаемого искажения D_1 по декодированному вектору $\mathbf{x}'(\mathbf{c})$. Используя выражение (9.19), получим:

$$\frac{\partial D_1}{\partial \mathbf{x}'(\mathbf{c})} = -\int_{-\infty}^{+\infty} d\mathbf{x} f_{\mathbf{X}}(\mathbf{x})\pi(\mathbf{c} - \mathbf{c}(\mathbf{x}))(\mathbf{x} - \mathbf{x}'(\mathbf{c})). \tag{9.21}$$

Отсюда, в свете (9.20) и (9.21), ранее изложенные условия 1 и 2 обобщенного алгоритма Ллойда можно модифицировать следующим образом [691].

Условие I. Для данного входного вектора $\mathbf{x}$ выбрать кодер $\mathbf{c}=\mathbf{c}(\mathbf{x})$, минимизирующий меру искажения:

$$D_2 = \int_{-\infty}^{\infty} d\mathbf{v}\pi(\mathbf{v}) \left\| \mathbf{x} - \mathbf{x}'(\mathbf{c}(\mathbf{x}) + \mathbf{v}) \right\|^2. \tag{9.22}$$

Условие II. Для данного кодера $\mathbf{c}$ вычислить восстановленный вектор $\mathbf{x}'(\mathbf{c})$, который удовлетворяет условию

$$\mathbf{x}'(\mathbf{c}) = \frac{\int_{-\infty}^{+\infty} d\mathbf{x} f_{\mathbf{X}}(\mathbf{x})\pi(\mathbf{c} - \mathbf{c}(\mathbf{x}))\mathbf{x}}{\int_{-\infty}^{+\infty} d\mathbf{x} f_{\mathbf{X}}(\mathbf{x})\pi(\mathbf{c} - \mathbf{c}(\mathbf{x}))}. \tag{9.23}$$

Уравнение (9.23) было получено приравниванием частной производной $\partial D_1 / \partial \mathbf{x}'(\mathbf{c})$ из (9.21) к нулю, после чего полученное уравнение решалось относительно $\mathbf{x}'(\mathbf{c})$.

Модель, показанную на рис. 9.5, можно рассматривать как частный случай модели, показанной на рис. 9.6. В частности, если в качестве функции плотности вероятности $\pi(\mathbf{v})$ шума $\mathbf{v}$ выбрать дельта-функцию Дирака $\delta(\mathbf{v})$, условия I и II сведутся к условиям 1 и 2 обобщенного алгоритма Ллойда.

Для того чтобы упростить условие I, предположим, что $\pi(\mathbf{v})$ является гладкой функцией аргумента $\mathbf{v}$. Тогда можно показать, что для аппроксимации второго порядка мера искажения D_2, описанная формулой (9.22), состоит из двух компонентов [691].

- Слагаемое *стандартного* искажения, определяемое квадратичной ошибкой искажения $||\mathbf{x} - \mathbf{x}'(\mathbf{c})||^2$.
- Слагаемое *искривления* (curvature), возникающее из модели шума $\pi(\mathbf{v})$.

Предполагая малость слагаемого искривления, условие I модели на рис. 9.6 может быть аппроксимировано условием 1 модели 9.5 без учета шума. Это, в свою очередь, сводит условие I к правилу ближайшего соседа, применяемому для кодирования.

Что же касается условия II, то его можно лучше интерпретировать, используя обучение методом стохастического спуска. В частности, из входного пространства $\mathbf{X}$ можно случайным образом выбрать вектор $\mathbf{x}$, используя множитель $\int d\mathbf{x} f_{\mathbf{X}}(\mathbf{x})$, и скорректировать реконструированный вектор $\mathbf{x}'(\mathbf{c})$ [691]:

$$\mathbf{x}'_{\text{new}}(\mathbf{c}) \leftarrow \mathbf{x}'_{\text{old}}(\mathbf{c}) + \eta\pi(\mathbf{c} - \mathbf{c}(\mathbf{x}))[\mathbf{x} - \mathbf{x}'_{\text{old}}(\mathbf{c})], \tag{9.24}$$

где η — параметр скорости обучения; $\mathbf{c}(\mathbf{x})$ — результат аппроксимации кодирования ближайшего соседа согласно условию 1. Уравнение коррекции (9.24) получено в результате анализа частной производной (9.21). Это уравнение можно применить ко всем $\mathbf{c}$, для которых

$$\pi(\mathbf{c} - \mathbf{c}(\mathbf{x})) > 0. \tag{9.25}$$

Процедуру, описываемую выражением (9.24), можно представить себе как способ минимизации меры искажения D_1 (9.21). Это значит, что выражения (9.23) и (9.24) имеют один и тот же тип, за исключением того факта, что (9.23) использует пакетный режим, а (9.24) — последовательный.

Уравнение коррекции (9.24) идентично (последовательному) алгоритму SOM (9.13), если принять во внимание соответствия, приведенные в табл. 9.1. Следовательно, можно утверждать, что обобщенный алгоритм Ллойда для векторной квантизации является пакетной версией алгоритма SOM с нулевым размером окрестности. Для такой окрестности $\pi(0) = 1$. Обратите внимание, что для получения обобщенного алгоритма Ллойда из пакетной версии алгоритма SOM *не нужно* делать никаких аппроксимаций, так как слагаемые искривления (и все слагаемые более высокого порядка) не вносят вклад в результат при *нулевой* ширине окрестности.

ТАБЛИЦА 9.1. Соответствия между алгоритмом SOM и моделью, показанной на рис. 9.6

Модель кодера-декодера, показанная на рис. 9.6	*Алгоритм SOM*
Кодер $\mathbf{c}(\mathbf{x})$	Наиболее подходящий нейрон $i(\mathbf{x})$
Восстановленный вектор $\mathbf{x}'(\mathbf{c})$	Вектор синаптических весов $\mathbf{w}_j$
Функция плотности вероятности $\pi(\mathbf{c}-\mathbf{c}(\mathbf{x}))$	Функция окрестности $h_{j,i(\mathbf{x})}$

В приведенном здесь обсуждении можно отметить следующие важные моменты.

- Алгоритм SOM представляет собой алгоритм векторного квантования, который осуществляет хорошую аппроксимацию входного пространства $\mathbf{X}$. Точка зрения, представляющая другой подход к выводу алгоритма SOM, представлена на примере выражения (9.24).
- Согласно этой точке зрения, функция окрестности $h_{j,i(x)}$ в алгоритме SOM имеет форму функции плотности вероятности. В [688] рассматривалась гауссова модель с нулевым средним, соответствующая шуму $\mathbf{v}$ в модели на рис. 9.6. Таким образом, у нас есть теоретические основания принятия гауссовой функции окрестности (9.4).

Пакетный алгоритм SOM[8] (batch SOM) является всего лишь другой формой выражения (9.23), в которой для аппроксимации интегралов в числителе и знаменателе в правой части уравнения используется суммирование. Обратите внимание, что в этой версии алгоритма SOM порядок представления сети входных образов не влияет на окончательную форму карты признаков и не существует необходимости в зависимости от времени параметра скорости обучения. Однако этот алгоритм все же требует использования функции окрестности.

Свойство 2. Топологический порядок. *Карта признаков* $\mathbf{\Phi}$, *полученная алгоритмом SOM, является топологически упорядоченной в том смысле, что пространственное положение нейронов в решетке соответствует конкретной области или признаку входного образа.*

Свойство топологической упорядоченности[9] является прямым следствием уравнения коррекции (9.13), которое перемещает вектор синаптических весов $\mathbf{w}_i$ победившего нейрона $i(\mathbf{x})$ в сторону входного вектора $\mathbf{x}$. Оно также имеет эффект перемещения вектора синаптических весов $\mathbf{w}_j$ ближайших нейронов вместе с весами

[8] В [571] были представлены экспериментальные результаты, показывающие, что пакетная версия алгоритма SOM быстрее интерактивной (on-line). Однако при использовании пакетной версии теряются адаптивные свойства этого алгоритма.

[9] Топологическое свойство самоорганизующейся карты может быть количественно измерено многими способами. Одна из таких мер, называемая *топологическим произведением* (topological product), описана в [104]. Ее можно использовать для сравнения корректности поведения различных карт признаков, имеющих разные размерности. Однако эта мера является количественной только в том случае, когда размерности решетки и входного пространства совпадают.

победившего нейрона $i(\mathbf{x})$. Таким образом, карту признаков $\mathbf{\Phi}$ можно представить в виде *гибкой* (elastic) или *виртуальной сети* (virtual net) с топологией одно- или двухмерной решетки, заданной выходным пространством $\mathbf{A}$, узлы которой имеют веса, соответствующие координатам во входном пространстве $\mathbf{X}$ [887]. Общую цель этого алгоритма можно сформулировать следующим образом.

Аппроксимировать входное пространство $\mathbf{X}$ *указателями или прототипами в форме векторов синаптических весов* $\mathbf{w}_j$ *таким образом, чтобы карта признаков* $\mathbf{\Phi}$ *обеспечивала верное представление важных признаков, характеризующих входные векторы* $\mathbf{x} \in \mathbf{X}$ *в терминах определенного критерия.*

Карту признаков $\mathbf{\Phi}$ обычно изображают во входном пространстве $\mathbf{X}$. В частности, все указатели (т.е. векторы синаптических весов) изображаются точками, а указатели соседних нейронов связываются линиями в соответствии с топологией решетки. Таким образом, используя линии для связывания двух указателей $\mathbf{w}_j$ и $\mathbf{w}_i$, мы отражаем тот факт, что нейроны j и i в решетке являются соседними.

Свойство 3. Соответствие плотности. *Карта признаков* $\mathbf{\Phi}$ *отражает вариации в статистиках распределения входного сигнала. Области во входном пространстве* $\mathbf{X}$*, из которого берутся векторы* $\mathbf{x}$ *с большей вероятностью, отображаются в гораздо большие области выходного пространства* $\mathbf{A}$ *и, таким образом, с большим разрешением, чем области в исходном пространстве* $\mathbf{X}$*, из которых берутся векторы* $\mathbf{x}$ *с меньшей вероятностью.*

Пусть $f_X(\mathbf{x})$ — многомерная функция распределения вероятности случайного входного вектора $\mathbf{x}$. Интеграл этой функции по всему пространству $\mathbf{X}$ должен равняться единице, т.е.

$$\int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x})d\mathbf{x} = 1.$$

Пусть $m(\mathbf{x})$ — *масштабирующий множитель* (magnification factor), определенный как количество нейронов в небольшой области $d\mathbf{x}$ входного пространства $\mathbf{X}$. Этот масштабирующий множитель, интегрированный по всему входному пространству $\mathbf{X}$, в результате дает общее количество l нейронов сети:

$$\int_{-\infty}^{\infty} m(\mathbf{x})d\mathbf{x} = l. \tag{9.26}$$

В алгоритме SOM, для того чтобы *точно соответствовать входной плотности* (match the input density exactly), требуется, чтобы

$$m(\mathbf{x}) \propto f_X(\mathbf{x}). \tag{9.27}$$

Это свойство означает, что если конкретная область входного пространства содержит часто встречающиеся возбудители, она должна быть представлена гораздо большей областью на карте признаков, чем та область входного пространства, в которой возбудители встречаются реже.

В общем случае в двумерной карте признаков масштабирующий множитель $m(\mathbf{x})$ нельзя считать простой функцией от функции плотности вероятности $f_{\mathbf{X}}(\mathbf{x})$ входного вектора $\mathbf{x}$. Это возможно только в случае одномерной карты признаков. В этом частном случае оказывается, что, в отличие от более раннего предположения [579], масштабирующий множитель $m(\mathbf{x})$ *не пропорционален* функции $f_{\mathbf{X}}(\mathbf{x})$. В литературе сообщается о двух совершенно различных результатах, зависящих от применяемого метода кодирования.

1. При *кодировании с минимальным искажением* (minimum-distortion encoding), в соответствии с которым сохраняются слагаемое искривления и все слагаемые более высокого порядка в измерении искажения (9.22), учитывающем влияние шума $\pi(\mathbf{v})$, этот метод приводит к следующему результату:

$$m(\mathbf{x}) \propto f_{\mathbf{X}}^{1/3}(\mathbf{x}), \tag{9.28}$$

   что идентично результату, полученному для стандартного алгоритма векторного квантования [688].

2. При *кодировании на основе метода ближайшего соседа* (nearest-neighbor coding), игнорирующего слагаемые искривления, как в стандартной форме алгоритма SOM, получается другой результат [886]:

$$m(\mathbf{x}) \propto f_{\mathbf{X}}^{2/3}(\mathbf{x}). \tag{9.29}$$

Прежнее утверждение о том, что кластер часто встречающихся входных возбуждений представляется большей областью на карте признаков, сохраняется, хотя и в несколько искаженной версии идеального условия (9.27).

Как правило (что подтверждено компьютерным моделированием), карта признаков, вычисленная алгоритмом SOM, имеет тенденцию в большей мере представлять области с низкой входной плотностью, чем области с высокой входной плотностью. Другими словами, алгоритм SOM не обеспечивает хорошее представление, описывающее входные данные[10].

---

[10] Неспособность алгоритма SOM дать правильное представленне распределения входиых даиных создала предпосылки для его модификации и появлення новых самоорганизующихся алгоритмов, корректиых по отиошению ко входному сигналу. В литературе упоминаются две такие модификации алгоритма SOM.

1. Модификация процесса коикуренции. В [254] использовалась некоторая форма памяти для отслеживания совокупной деятельности отдельных нейронов в решетке. В частности, был добавлен механизм "совести" (conscience) для смещеиия процесса коикурентного обучения алгоритма SOM. Это было сделано так, что

Свойство 4. Выбор признаков. *Для данных из входного пространства с нелинейным распределением самоорганизующаяся карта для аппроксимации исследуемого распределения способна извлечь набор наилучших признаков.*

Это свойство является естественной кульминацией свойств 1–3. Оно заставляет вспомнить идею анализа главных компонентов, которая рассматривалась в предыдущей главе. Однако это свойство имеет одно важное отличие, которое показано на рис. 9.7. На рис. 9.7, *а* показано двумерное распределение с нулевым средним точек данных, полученных в результате линейного отображения входа на выход, искаженное аддитивным шумом. В этом случае анализ главных компонентов работает превосходно. Он позволяет определить, что наилучшее описание "линейного" распределения на рис. 9.7, *а* задается прямой линией (т.е. одномерной гиперплоскостью), которая проходит через начало координат и следует параллельно собственному вектору, ассоциированному с наибольшим собственным значением матрицы корреляции данных. Теперь рассмотрим другую ситуацию (см. рис. 9.7, *б*), являющуюся результатом нелинейного отображения входа на выход, искаженного аддитивным шумом с нулевым средним значением. В этом, втором, случае аппроксимация прямой линией, вычисленная с помощью анализа главных компонентов, не способна обеспечить приемлемое описание данных. С другой стороны, использование самоорганизующейся карты, построенной на одномерной решетке нейронов, обходит эту проблему аппроксимации благодаря своему свойству упорядочивания топологии. Эта, последняя, аппроксимация показана на рис. 9.7, *б*.

Переходя к строгой терминологии, можно утверждать, что самоорганизующиеся карты признаков осуществляют *дискретную* аппроксимацию так называемых *главных*

каждый нейрои, иезависимо от его расположения в решетке, имел шанс выиграть соревиование с вероятностью, близкой к идеальной (1/l), где l — общее количество нейронов. Описание алгоритма SOM с учетом "совести" представлено в задаче 9.8.

2. *Модификация адаптивного процесса.* В этом подходе правило коррекции для настройки вектора весов каждого из нейронов, попадающих в функцию окрестиости, модифицировалось для управления масштабирующими (magnification) свойствами карты признаков. В [105] было показано, что благодаря добавлению в правило коррекции иастраиваемого параметра размера шага удалось обеспечить хорошее представление картой призиаков входного распределения. Авторы [645] следовали по тому же пути и предложили две модификации алгоритма SOM.
   - Правило коррекции модифицировалось для извлечения прямой зависимости между входным вектором $\mathbf{x}$ и вектором весов $\mathbf{w}_j$ рассматриваемого иейроиа j.
   - Разбиение Вороного было замеиено равноцеиным разбиением, специальио созданным для разделяемых входных распределений.

   Эта вторая модификация позволила алгоритму SOM осуществлять слепое разделение входных сигналов (blind source separation). (Слепое разделеиие сигиалов вкратце освещалось в главе 1 и будет более полно рассматриваться в главе 10.)

Эти модификации были построеиы на той или иной форме алгоритме SOM. В [652] был предпринят совершенно иной подход. В частности, общее правило обучения для формирования топографической карты содержало максимизацию взаимной информации (mutual information) между выходным сигналом и сигнальной частью входа, искажениой аддитивиым шумом. (Понятие взаимной ииформации, берущее свои корни в теории информации Шеииона, опнсывается в главе 10.) Модель, предложеиная в этой работе, в результате содержала распределеиие нейронов, в точности соответствующее входиому распределению. Использование к самоорганизующемуся формированию топографической карты подхода со стороны теории информации также описано в [1081], [1082].

кривых[11] (principal curve) или *главных поверхностей* (principal surface) [430]. Таким образом, они могут рассматриваться как нелинейное обобщение анализа главных компонентов.

9.6. Компьютерное моделирование

Двумерная решетка, полученная на основе двумерного распределения

Теперь проиллюстрируем работу алгоритма SOM с помощью компьютерного моделирования. Будем изучать сеть, состоящую из 100 нейронов, упорядоченных в форме двумерной решетки размером 10×10. Эта сеть обучается на двумерных входных векторах $\mathbf{x}$, элементы x_1 и x_2 которых равномерно распределены в областях $\{(-1 < x_1 < +1); (-1 < x_2 < +1)\}$. Для инициализации сети синаптические веса выбираются случайным образом.

На рис. 9.8 показаны три этапа обучения сети представлению входного распределения. На рис. 9.8, *а* показано равномерное распределение данных, использованных для обучения карты признаков. На рис. 9.8, *б* показаны исходные значения случайно выбранных синаптических весов. На рис. 9.8, *в*, *г* показаны значения векторов синаптических весов после завершения этапов упорядочивания и сходимости. Синаптические веса на этих рисунках показаны точками. Линии, показанные на рис. 9.8, соединяют соседние нейроны (вдоль строк и столбцов) сети.

Результаты, которые мы видим на рис. 9.8, демонстрируют этапы упорядочивания и сходимости, которые характеризуют процесс обучения по алгоритму SOM. Во время этапа упорядочивания карта напоминает форму сетки (см. рис. 9.8, *в*). К концу этого этапа нейроны отображаются в правильном порядке. Во время этапа сходимости сетка пытается "накрыть" все входное пространство. В конце второго этапа (см. рис. 9.8, *г*) статистическое распределение нейронов на карте приближается к распределению входных векторов (за исключением некоторых граничных эффектов). Сравнивая конечное состояние карты признаков (см. рис. 9.8, *г*) с равномерным распределением входных сигналов (см. рис. 9.8, *а*), мы видим, что точная настройка карты, выполненная на этапе сходимости, выявила отдельные локальные неравномерности, присутствующие во входном распределении.

---

[11] Связь между алгоритмом SOM и главными (principal) кривыми обсуждается в [186], [891]. Алгоритм отыскания главных кривых состоит из двух шагов [430].

1. *Проекция*. Для каждой точки данных находится ее ближайшая проекция на кривую (т.е. самая ближайшая точка на кривой).
2. *Условное ожидание* (conditional exceptation). Применяем сглаживание графика разброса к проектируемым значениям вдоль кривой. В этой процедуре рекомендуется начинать с большого диапазона и постепенно уменьшать его.

Эти два шага аналогичны векторному квантованию и процедуре "отжига" окрестности, осуществляемым в алгоритме SOM.

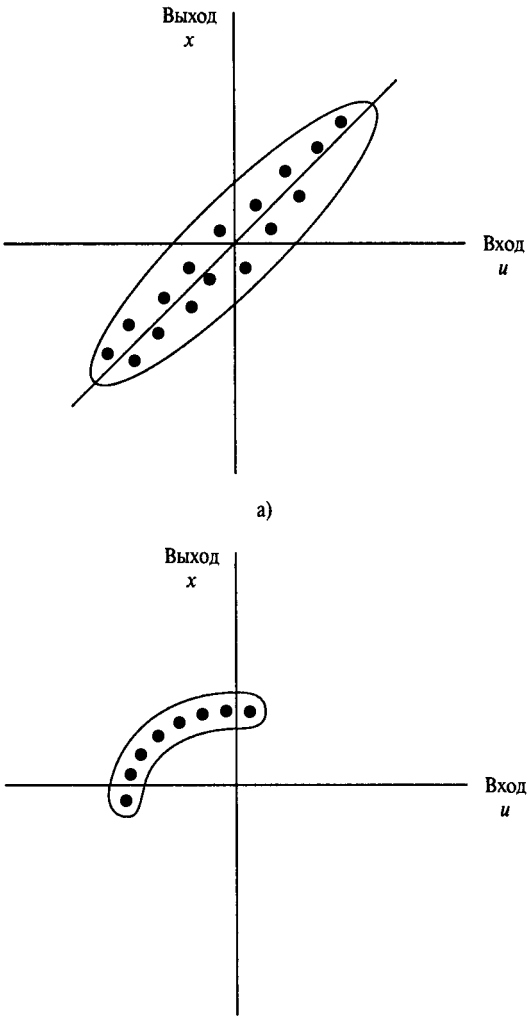


Рис. 9.7. Двумерное распределение, полученное в результате линейного отображения входа на выход (а); двумерное распределение, полученное в результате нелинейного отображения входа на выход (б)

Свойство топологического упорядочивания алгоритма SOM хорошо продемонстрировано на рис. 9.8, *г*. В частности, видно, что алгоритм (после сходимости) извлек топологию распределения входного сигнала. В результатах компьютерного моделирования (см. рис. 9.8) как входное пространство **X**, так и выходное пространство **A** было двумерным.

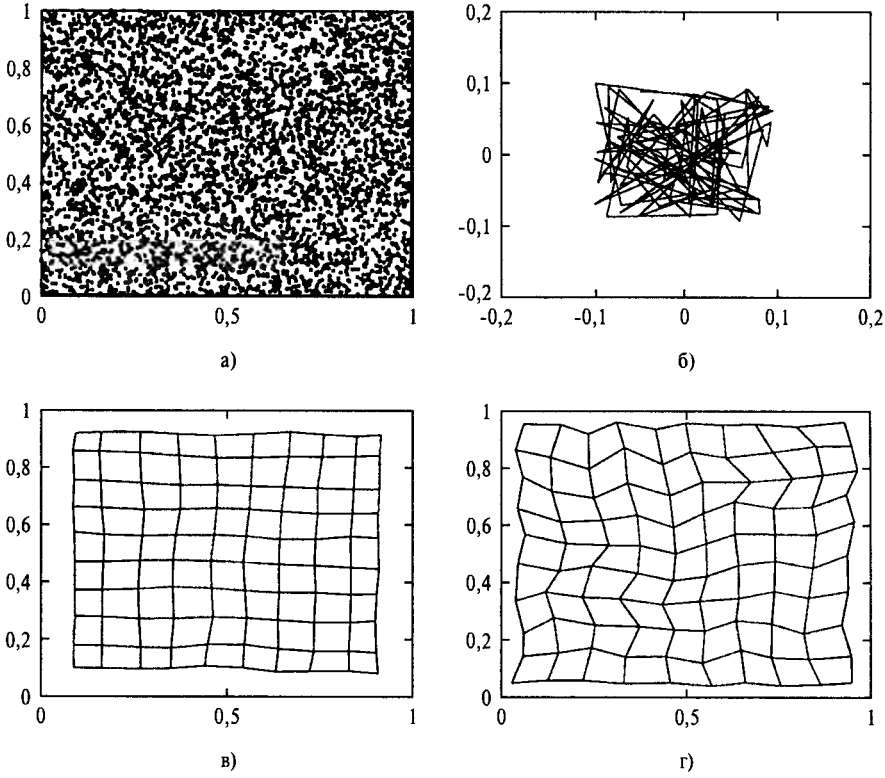


Рис. 9.8. Распределение входных данных (а); начальное состояние двумерной решетки (б); состояние решетки в конце этапа упорядочивания (в); состояние решетки в конце этапа сходимости (г)

Одномерная решетка на основе двумерного распределения

Теперь рассмотрим случай, когда размерность входного пространства **X** больше размерности выходного пространства **A**. Несмотря на это расхождение, карта признаков **Ф** часто способна сформировать топологическое представление входного распределения. На рис. 9.9 показаны три этапа эволюции карты признаков, инициализированной на рис. 9.9, *б* и обучаемой на входных данных, взятых из равномерного распределения внутри квадрата (см. рис. 9.9, *а*). Вычисления осуществлялись на одномерной решетке, состоящей из 100 нейронов. На рис. 9.9, *в*, *г* показаны карты признаков после завершения этапов упорядочивания и сходимости соответственно. На этих рисунках видно, что карта признаков, вычисленная алгоритмом, не может корректно представить равномерное заполнение квадрата достаточно плотно и дать достаточно хорошую аппроксимацию двумерного входного пространства **X**. Кривая аппроксимации, показанная на рис. 9.9, *г*, является *кривой Пеано* (Peano curve) [573]. Операция, проведенная в этом эксперименте и представленная картой признаков на рис. 9.9 (где входное пространство **X** представлено его проекцией на пространство более низкой размерности **A**), называется *снижением размерности* (dimensionality reduction).

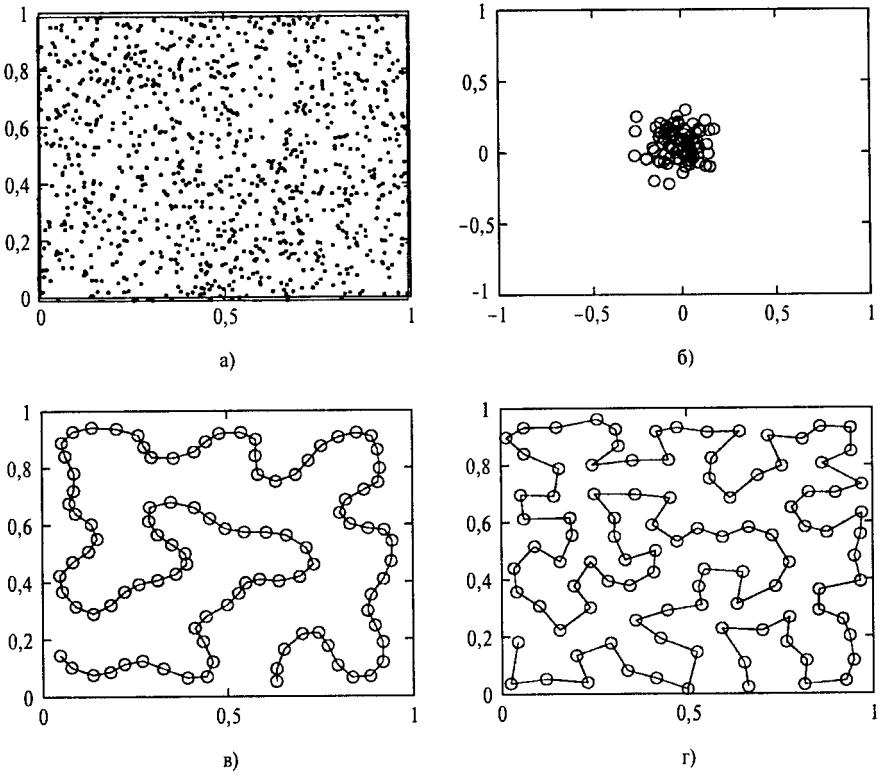


Рис. 9.9. Распределение входных данных (а); начальное состояние двумерной решетки (б); состояние решетки в конце этапа упорядочивания (в); состояние решетки в конце этапа сходимости (г)

Описание параметров моделирования

На рис. 9.10 представлены детали динамики функции окрестности $h_{j,i}(n)$ и параметра скорости обучения $\eta(n)$ с течением времени (т.е. с количеством эпох) при моделировании отображения на одномерную решетку. Параметр функции окрестности $\sigma(n)$, показанный на рис. 9.10, *а*, начинает работу из исходного состояния $\sigma_0 = 18$ и убывает практически до единицы за 1000 итераций этапа упорядочивания. В течение того же этапа параметр скорости обучения $\eta(n)$ начинает работу со значения 0,1 и уменьшается до значения 0,037. На рис. 9.10, *в* показано изначальное гауссово распределение нейронов вокруг победившего нейрона, расположенного в центре одномерной решетки. На рис. 9.9, *г* показана форма функции окрестности в конце этапа упорядочивания. Во время этапа сходимости (5000 итераций) параметр скорости обучения равномерно уменьшался от 0,037 до 0,001. В течение того же этапа функция окрестности сократилась практически до нуля.

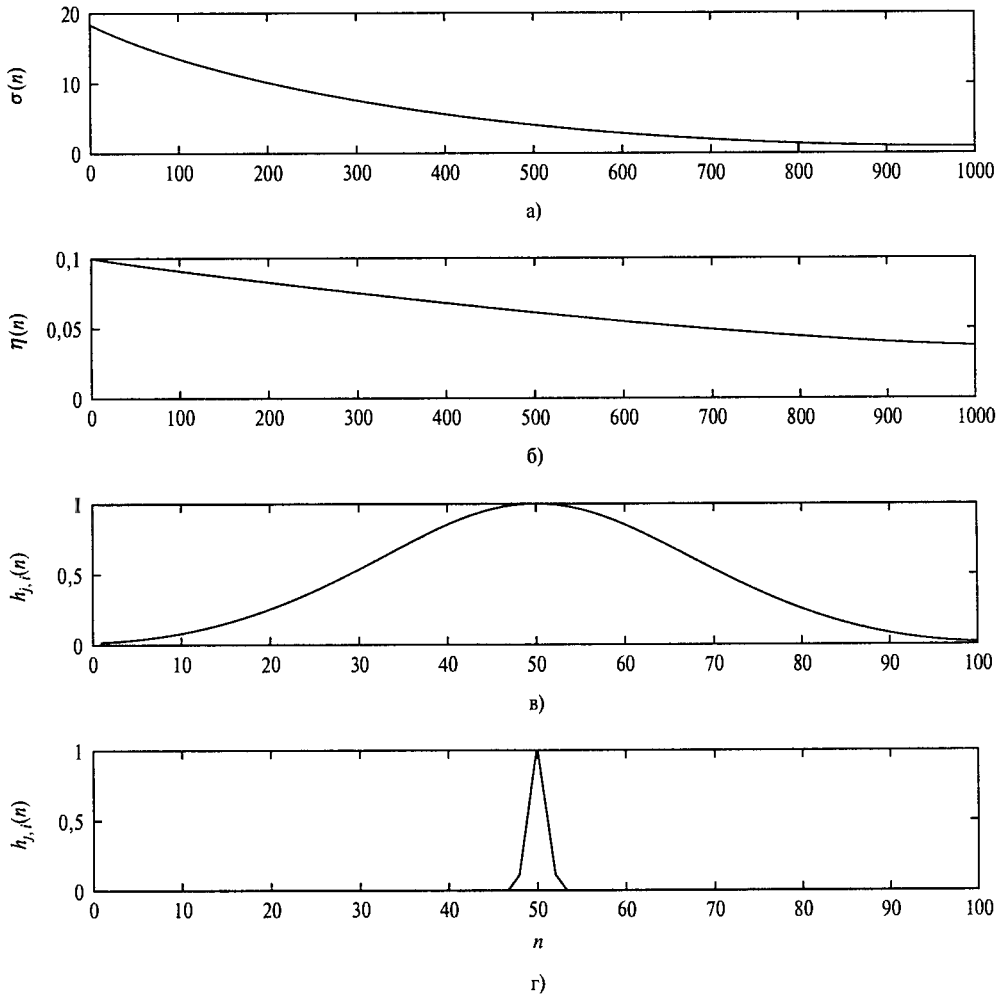


Рис. 9.10. Экспоненциальное убывание параметра $\sigma(n)$ функции окрестности (а); экспоненциальное убывание параметра $\eta(n)$ (б); изначальная форма гауссовой функции окрестности (в); форма функции окрестности по завершении фазы упорядочивания (т.е. в начале этапа сходимости) (г)

Спецификации этапов упорядочивания и сходимости при компьютерном моделировании для случая двумерной решетки (см. рис. 9.8) аналогичны использованным для одномерной, за исключением того факта, что функция окрестности теперь тоже стала двумерной. Параметр $\sigma(n)$ начал работу со значения $\sigma_0 = 3$, за 1000 итераций уменьшившись до $0,75$. На рис. 9.11 показано начальное состояние гауссовой функции окрестности $h_{j,i}$ для $\sigma_0 = 3$ и победившего нейрона, центрированного в точке (7; 8) двумерной решетки размером 10×10 нейронов.

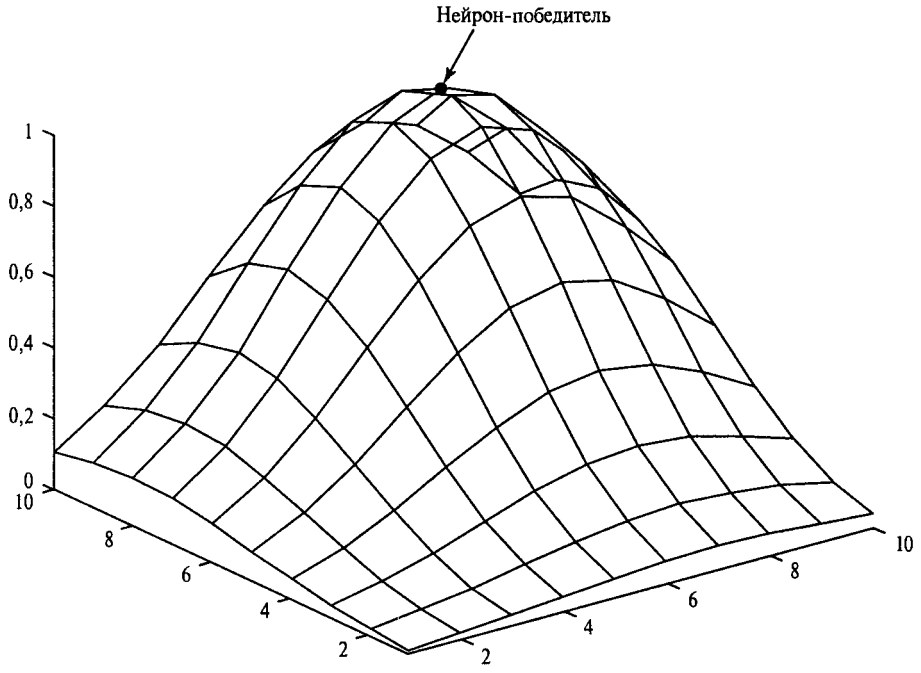


Рис. 9.11. Начальное условие для двумерной гауссовой функции окрестности с центром в нейроне-победителе, расположенном в точке (7; 8) двумерной решетки размером 10 × 10 нейронов

9.7. Квантование вектора обучения

Векторное квантование (vector quantization), о котором мы говорили в разделе 9.6, представляет собой прием использования структуры входных векторов для сжатия данных [346]. В частности, входное пространство подразделяется на множество четких областей, для каждой из которых определяется вектор воспроизведения. Когда устройству квантования представляется новый входной вектор, в первую очередь определяется область, к которой принадлежит данный вектор, а после этого создается представление через вектор воспроизведения данной области. Таким образом, используя некодированную версию восстанавливаемого вектора для хранения или передачи исходного вектора, можно добиться значительной экономии пространства хранения или мощности канала передачи данных за счет привнесения некоторых искажений. Множество возможных векторов воспроизведения называется *кодовой книгой* (code book) устройства квантования, а его отдельные члены — *кодовыми словами* (code word).

Устройство векторного квантования, обладающее минимальным искажением при кодировании, называется *квантователем Вороного*, или *квантователем на основе ближайшего соседа* (nearest-neighbor quantizer), а *ячейками Вороного* называют множество точек во входном пространстве, которые соответствуют подразделению этого

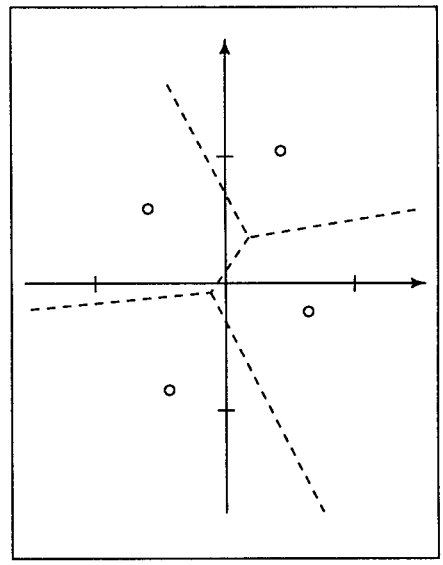


Рис. 9.12. Диаграмма Вороного, содержащая 4 ячейки ([379], приведена с разрешения IEEE)

пространства в соответствии с *правилом ближайшего соседа*, основанным на Евклидовой метрике [346]. На рис. 9.12 показан пример входного пространства, разделенного на четыре ячейки Вороного, с соответствующими векторами воспроизведения (или векторами Вороного). Каждая из этих ячеек содержит те точки входного пространства, которые расположены ближе всего к вектору Вороного.

Алгоритм SOM обеспечивает приближенный метод вычисления векторов Вороного без учителя. При этом аппроксимация определяется векторами синаптических весов нейронов на карте признаков. Это практически повторяет описание свойства 1 алгоритма SOM, о котором речь шла в разделе 9.6. Расчет карты признаков, таким образом, можно рассматривать как первый из двух этапов адаптивного решения задачи классификации (рис. 9.13). На втором этапе в качестве механизма точной подстройки карты признаков проводится квантование вектора обучения.

Квантизация вектора обучения[12] (learning vector quantization — LVQ) — это прием обучения с учителем, который использует информацию о классе для небольшого смещения вектора Вороного и, таким образом, для улучшения качества областей решений классификатора. Входной вектор $\mathbf{x}$ случайно выбирается из входного пространства. Если метки класса входного вектора $\mathbf{x}$ и вектора Вороного $\mathbf{w}$ согласуются, последний смещается в направлении первого. Если же метки классов этих векторов не согласуются, вектор Вороного $\mathbf{w}$ смещается в сторону, противоположную входному вектору $\mathbf{x}$.

---

[12] Идея квантования вектора обучения принадлежит Кохонену [578]. Три версии этого алгоритма были описаны в его работах [568], [574]. Версия, предлагаемая в разделе 9.7, является первой из них и в работах Кохонена обозначается как LVQ1.

Алгоритм квантования вектора обучения является алгоритмом стохастической аппроксимации. В работе [89] исследовались свойства сходимости этого алгоритма. При этом использовался подход ODE (обычных дифференциальных уравнений), который описывался в главе 8.

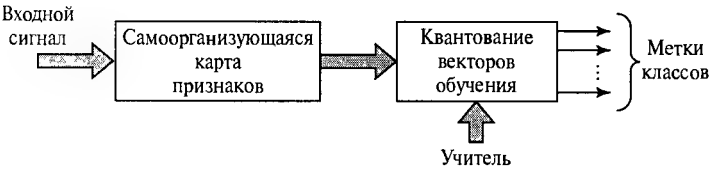


Рис. 9.13. Блочная диаграмма адаптивной классификации множеств, использующей самоорганизующуюся карту признаков и квантизацию вектора обучения

Пусть $\{\mathbf{w}_j\}_{j=1}^{l}$ — множество векторов Вороного, а $\{\mathbf{x}_i\}_{i=1}^{N}$ — множество входных векторов (наблюдений). Предполагается, что количество входных векторов намного превосходит количество векторов Вороного, что, как правило, происходит на практике. Алгоритм квантования векторов обучения (LVQ) можно описать следующим образом.

1. Предположим, что вектор Вороного $\mathbf{w}_c$ является самым близким к входному вектору $\mathbf{x}_i$. Обозначим символом $\mathbf{C}_{\mathbf{w}_c}$ класс, ассоциируемый с вектором Вороного $\mathbf{w}_c$, а символом $\mathbf{C}_{\mathbf{x}_i}$ — метку класса входного вектора $\mathbf{x}_i$. Вектор Вороного $\mathbf{w}_c$ корректируется следующим образом.
   - Если $\mathbf{C}_{\mathbf{w}_c} = \mathbf{C}_{\mathbf{x}_i}$, то

$$\mathbf{w}_c(n+1) = \mathbf{w}_c(n) + \alpha_n[\mathbf{x}_i - \mathbf{w}_c(n)], \tag{9.30}$$

     где $0 < \alpha_n < 1$.
   - Если $\mathbf{C}_{\mathbf{w}_c} \neq \mathbf{C}_{\mathbf{x}_i}$, то

$$\mathbf{w}_c(n+1) = \mathbf{w}_c(n) - \alpha_n[\mathbf{x}_i - \mathbf{w}_c(n)]. \tag{9.31}$$

2. Остальные векторы Вороного не изменяются.

Константу обучения α_n лучше сделать монотонно убывающей с увеличением числа итераций. Например, начальным значением α_n может служить 0,1 или меньшее. Затем эта константа может убывать линейно по n. После нескольких проходов по входным данным векторы Вороного, как правило, сходятся. После этого обучение считается завершенным. Однако если метод применяется без должного внимания, можно столкнуться с определенными трудностями.

9.8. Компьютерное моделирование: адаптивная классификация множеств

В задаче классификации множеств первым и наиболее важным шагом является *извлечение признаков* (feature selection), которое обычно выполняется без учителя. Целью этого шага является выбор обоснованно малого количества признаков, в которых

сконцентрирована самая существенная информация о входных (классифицируемых) данных. Для извлечения признаков лучше всего подходит самоорганизующаяся карта, ввиду ее свойства 4 (см. раздел 9.5), особенно если входные данные формируются нелинейным процессом.

Вторым шагом в задаче классификации множеств является сама *классификация*, в которой признакам, выделенным на первом шаге, назначаются разные классы. Несмотря на то что и самоорганизующаяся карта может самостоятельно осуществлять классификацию, все же для повышения производительности рекомендуется дополнить ее какой-нибудь схемой обучения с учителем. Комбинация самоорганизующейся карты со схемой обучения с учителем формирует базис *адаптивной классификации множеств* (adaptive pattern classification), которая является *гибридной* по своей сущности.

Такой гибридный подход к классификации может иметь различные формы, в зависимости от того, как реализована схема обучения с учителем. Одна из простейших схем использует квантователь вектора обучения, описанный в предыдущем разделе. Исходя из сказанного, получим двухступенчатый адаптивный классификатор (см. рис. 9.13).

В этом эксперименте снова вернемся к задаче классификации двух двумерных пересекающихся множеств с гауссовым распределением, которые можно пометить как классы $\mathbf{C}_1$ и $\mathbf{C}_2$. Эта задача более подробно описывалась в главе 4, где для ее решения использовался многослойный персептрон, обучаемый методом обратного распространения. График распределения данных, используемых в эксперименте, показан на рис. 4.13.

На рис. 9.14, *а* показана двумерная карта признаков, состоящая из решетки размером 5×5 нейронов после завершения работы алгоритма SOM. Эта карта была маркирована, и каждый из нейронов был поставлен в соответствие одному из классов, в зависимости от того, как он среагировал на тестовые данные, взятые из входного распределения. На рис. 9.14, *б* показана граница решений, сформированная самой картой признаков.

На рис. 9.14, *в* показана модифицированная карта признаков, настроенная в процессе обучения с учителем на основе алгоритма LQV. На рис. 9.14, *г* показана граница решений, полученная в результате совместного действия алгоритмов SOM и LQV. Сравнивая последние два рисунка с их двойниками, 9.14, *а* и 9.14, *б*, мы видим качественный эффект применения алгоритма LQV.

В табл. 9.2 представлены эффективности классификации для карты признаков, действующей самостоятельно, и для совмещения алгоритмов SOM и LQV. Представленные здесь результаты получены из 10 независимых пробных эксперимента, в каждой из которых использовалось по 30000 примеров тестовых данных. В каждом эксперименте наблюдалось повышение качества классификации при применении алгоритма LQV. Средний показатель производительности для карты признаков, дей-

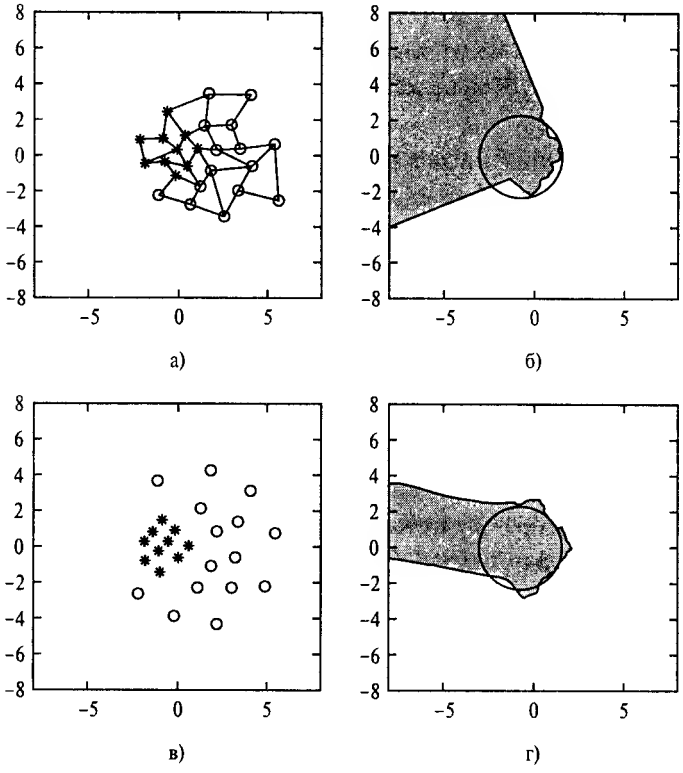


Рис. 9.14. Самоорганизующаяся карта после расстановки меток (а). (б) Граница решений, построенная картой (а). Маркированная карта после квантования векторов обучения (в). (г) Граница решений, построенная на карте (в)

ствующей самостоятельно, составил 79,61%, а для комбинации двух алгоритмов — 80,52%, т.е. среднее улучшение качества классификации составило 0,91%. Для сравнения вспомним, что производительность оптимального классификатора Байеса для этого эксперимента составила 81,51%.

9.9. Иерархическое квантование векторов

При обсуждении свойства 1 самоорганизующейся карты признаков в разделе 9.6 мы особо отметили, что она тесно связана с обобщенным алгоритмом Ллойда для векторного квантования. Векторное квантование является формой сжатия с *потерей данных*. Это значит, что в результате сжатия теряется часть информации, содержащейся в данных. Своими корнями сжатие данных уходит в теорию информации Шеннона (Shennon), еще называемую *теорией уровня искажения* (rate distortion theory) [221]. В свете нашей дискуссии об иерархической векторного квантования будет уместным начать с утверждения, явившегося фундаментальным результатом теории уровня искажения [379].

ТАБЛИЦА 9.2. Эффективность классификации (в процентах) в компьютерном эксперименте по классификации двух пересекающихся гауссовых распределений с помощью решетки размером 5×5

Попытка	*Самостоятельная классификация картой признаков*	*Классификация комбинацией карты признаков и LVQ*
1	79,05	80,18
2	79,79	80,56
3	79,41	81,17
4	79,38	79,84
5	80,30	80,43
6	79,55	80,36
7	79,79	80,86
8	78,48	80,21
9	80,00	80,51
10	80,32	81,06
Среднее	79,61%	80,52%

Наилучшего сжатия данных всегда можно добиться с помощью кодирования не векторов, а скаляров, даже если источник данных не имеет памяти (т.е. является последовательностью независимых случайных переменных), или если система сжатия данных имеет память (т.е. действия кодировщика зависят от своих предыдущих входов или выходов).

За этим фундаментальным результатом стоят обширные исследования в области векторного квантования [346].

Однако алгоритмы обычной векторной квантизации требуют довольного большого объема вычислений, что ограничивает их практическое использование. Большую часть времени в алгоритме векторной квантизации занимает операция кодирования. При кодировании входной вектор должен сравниваться с каждым из векторов кодирования в кодовой книге, для того чтобы найти тот, который обеспечивает минимальное искажение. Например, если кодовая книга содержит N векторов кодирования, время, затраченное на кодирование, будет иметь порядок N. Исходя из этого, с увеличением N оно будет возрастать. В [690] описано *многоступенчатое иерархическое векторное квантование* (multistage hierarchical vector quantizer), которое значительно увеличивает скорость кодирования. Эта схема не является простым деревом поиска в кодовой книге. В ней реализован кардинально новый принцип. Многоступенчатый иерархический векторный квантователь раскладывает общую операцию векторного квантования на множество подопераций, каждая из которых требует крайне малого объема вычислений. Желательно, чтобы такое разложение сводилось к одной процедуре поиска на подоперацию. Мудро используя алгоритм SOM для обучения каждой из подопераций, можно добиться малой потери точности (до доли децибела) при одновременном повышении скорости вычислений.

Рассмотрим два векторных квантователя, VQ_1 и VQ_2, где первый передает свой выход на второй. Выход второго квантователя является окончательной кодированной версией исходного входного сигнала, примененного к VQ_1. При осуществлении квантования совершенно нежелательно, чтобы VQ_2 отвергал какую-либо информацию. Поскольку рассматривается квантователь VQ_1, исключительный эффект от VQ_2 заключается в искажении выхода VQ_1. Таким образом, получается, что для VQ_1 больше подходит алгоритм SOM, который учитывает искажение сигнала, налагаемое квантователем VQ_2 [690]. Для того чтобы использовать обобщенный алгоритм Ллойда для обучения квантователя VQ_2, нужно предположить, что выход VQ_2 не будет искажен до проведения восстановления. В этом случае не потребуется вводить зашумленную модель (для выхода VQ_2) и ассоциированную с ней функцию окрестности конечной ширины.

Этот эвристический аргумент можно обобщить на многоступенчатый векторный квантователь. Каждый шаг должен учитывать искажения, привнесенные всеми предыдущими шагами, и моделировать их как шум. Поэтому для обучения всех ступеней квантователя (за исключением последней, для которой применяется обобщенный алгоритм Ллойда) используется алгоритм SOM.

Иерархическое векторное квантование является частным случаем многоступенчатого векторного квантования [690]. В качестве иллюстрации рассмотрим квантование входного вектора размерности 4×1:

$$\mathbf{x} = [x_1, x_2, x_3, x_4]^T.$$

На рис. 9.15, *а* показан одноступенчатый векторный квантователь для $\mathbf{x}$. В качестве альтернативы используется двухступенчатый иерархический квантователь, представленный на рис. 9.15, *б*. Разительное отличие между двумя этими схемами состоит в том, что размерность входа квантователя на рис. 9.15, *а* равна четырем, в то время как квантователя на рис. 9.15, *б* — двум. Соответственно квантователь, представленный на рис. 9.15, *б*, требует для своей работы существенно меньшей *справочной таблицы* (look-up table) и является более простым для реализации, чем квантователь, показанный на рис. 9.15, *а*. В этом и лежит главное преимущество иерархического квантователя перед обычным.

В [690] продемонстрирована эффективность многоступенчатого иерархического векторного квантователя применительно к различным стохастическим рядам. При этом потери в точности были минимальными. На рис. 9.16 воспроизведены результаты, полученные в этой работе, для случая коррелированного гауссова шума, генерируемого с помощью *авторегрессионой модели первого порядка* (first-order autoregressive model):

$$x(n+1) = \rho x(n) + v(n), \tag{9.32}$$

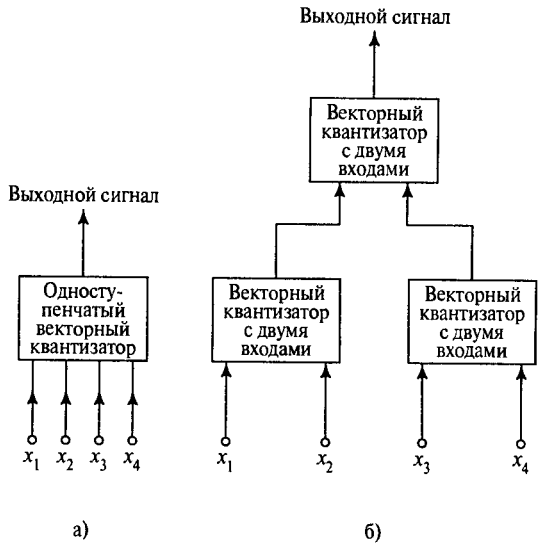


Рис. 9.15. Одноступенчатый векторный квантователь с четырехмерным входным сигналом (а); двухступенчатый иерархический векторный квантователь, использующий двухвходовые векторные квантователи (б) (из [690])

где ρ — коэффициент авторегрессии; $v(n)$ — независимая и равномерно распределенная гауссова случайная переменная с нулевым средним и единичной дисперсией. Исходя из этого, можно показать, что $x(n)$ характеризуется следующим образом:

$$E[x(n)] = 0, \tag{9.33}$$

$$E[x^2(n)] = \frac{1}{1-\rho^2}, \tag{9.34}$$

$$\frac{E[x(n+1)x(n)]}{E[x^2(n)]} = \rho. \tag{9.35}$$

Таким образом, ρ можно рассматривать как *коэффициент корреляции* (correlation coefficient) временного ряда $\{x(n)\}$. Чтобы инициировать генерацию этого временного ряда в соответствии с (9.32), для $x(0)$ использована гауссова случайная переменная с нулевым средним и дисперсией $1/(1-\rho^2)$, а в качестве коэффициента корреляции выбрана величина $\rho = 0,85$.

Для векторной квантизации использовался иерархический кодер с четырехмерным входным пространством, подобный показанному на рис. 9.15, *б*. Для ряда авторегрессии $\{x(n)\}$ потребовалась симметрия преобразования, подразумевающая использование только двух отдельных справочных таблиц. Размер каждой из этих таблиц экспоненциально зависел от количества битов входного сигнала и линейно от количества битов выходного сигнала. Во время обучения для правильного вычисления коррекций (9.24) потребовалось большое количество битов для представления чисел, поэтому во время обучения справочные таблицы не использовались. Однако после

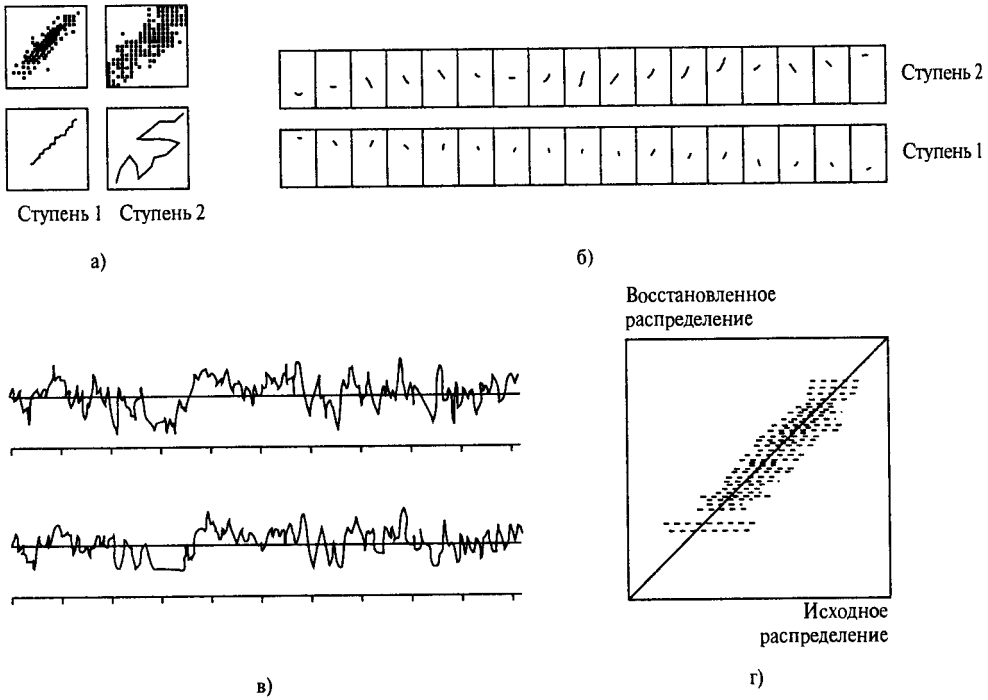


Рис. 9.16. Результаты двухступенчатого кодирования/декодирования для коррелированного гауссова шума. Коэффициент корреляции $\rho = 0,85$ *(из [690])*

завершения обучения количество битов могло быть уменьшено до обычного уровня, после чего потребовались записи справочных таблиц. Для кодера, показанного на рис. 9.15, *б*, входные образы аппроксимировались с использованием четырех битов на образ. Для всех ступеней кодера использовалось $N = 17$ векторов кодирования, так что количество выходных битов в каждой из справочных таблиц приблизительно равнялось четырем. Таким образом, размер адресного пространства для таблиц обеих ступеней составил 256 ($=2^{4+4}$). Это говорит о том, что общие требования к памяти для представления таблиц были крайне скромными.

На рис. 9.16 показаны результаты кодирования/декодирования элементов $x(n)$. В нижней части рис. 9.16, *а* векторы кодирования для каждой из двух ступеней показаны как кривые, помещенные в двумерное входное пространство. В верхней части рис. 9.16, *а* показаны оценки соответствующих матриц *соответствия* (co-occurence) размером 16×16. На рис. 9.16, *б* в виде фрагментов временного ряда представлены следующие элементы.

- Вектор кодирования, вычисленный первой ступенью кодера.
- Вычисленный второй ступенью вектор восстановления, который минимизирует среднеквадратическое искажение при одновременном сохранении всех остальных переменных.

На рис. 9.16, *в* представлено 512 примеров из исходного временного ряда (верхняя кривая) совместно с реконструированными примерами (нижняя кривая). Эти данные взяты из выхода последней ступени кодировщика. Горизонтальная шкала на рис. 9.16, *в* равна половине шкалы на рис. 9.16, *б*. На рис. 9.16, *г* представлена матрица соответствия, составленная из пар примеров: исходных и соответствующих им восстановленных. Ширина полосы на рис. 9.16, *г* показывает степень искажения, произведенного иерархической векторной квантизацией.

Исследуя графики на рис. 9.16, *в*, можно увидеть, что восстановленная информация является достаточно хорошим представлением исходного временного ряда, за исключением того, что отдельные положительные и отрицательные пики срезаны. Согласно [690], нормализованное среднеквадратическое искажение составило 0,15, что практически не хуже (меньше на 0,05 дБ), чем 8,8 дБ, полученные с использованием одноступенчатого блочного кодера с четырьмя входами, использующего по одному биту на пример [510].

9.10. Контекстные карты

Существуют два фундаментально отличных метода визуализации самоорганизующихся карт признаков. Первый из этих методов состоит в построении гибкой сети, в которой векторы синаптических весов являются указателями, направленными от соответствующих нейронов во входное пространство. Этот метод визуализации особенно полезен для правильного отображения свойства топологического упорядочивания алгоритма SOM. Это было продемонстрировано на примере компьютерного моделирования, приведенного в разделе 9.6.

Во втором методе визуализации нейронам в двумерной решетке (представляющим выходной слой сети) назначаются метки классов, в зависимости от того, как каждый из примеров (не встречавшихся ранее) возбудил конкретный нейрон в самоорганизующейся сети. В результате этой второй ступени моделирования нейроны в двумерной решетке разбиваются на некоторое количество *когерентных областей* (coherent region). Здесь под когерентностью понимается то, что каждая из групп нейронов представляет обособленное множество непрерывных символов или меток [888]. Это подразумевает, что при восстановлении хорошо упорядоченной карты использовались правильные условия.

Для примера рассмотрим множество данных, представленных в табл. 9.3. В этой таблице приведены характеристики отдельных животных. На основании данных каждого из столбцов таблицы можно составить описание некоторого животного. При этом значение 1 подразумевает наличие, а 0 — отсутствие одного из 13 свойств, перечисленных в таблице слева. Отдельные атрибуты (например, “2 ноги” и “4 ноги”) коррелируют, в то время как остальные — нет. Для каждого из представленных животных можно составить *характеристический код* (attribute code) $\mathbf{x}_a$, состоящий из 13 элемен-

ТАБЛИЦА 9.3. Названия животных и их характеристики

Животное		*Гусь*	*Кулик*	*Утка*	*Гусь*	*Сова*	*Ястреб*	*Орел*	*Лиса*	*Собака*	*Волк*	*Кошка*	*Тигр*	*Лев*	*Лошадь*	*Зебра*	*Корова*
Размер	*Маленький*	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	Средний	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	Крупный	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Имеет	*2 ноги*	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 ноги	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Шерсть	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Копыта	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	Гриву	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	Перья	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Любит	*Охоту*	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	Бегать	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	Летать	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	Плавать	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

тов. Само животное определяется *символьным кодом* (symbolic code) $\mathbf{x}_s$, по которому никак нельзя собрать информацию о сходствах и различиях между отдельными животными. Например, в данном примере $\mathbf{x}_s$ представляет собой вектор, k-й элемент (соответствующий номеру животного в списке) которого имеет некоторое значение a; при этом все остальные элементы равны нулю. Параметр a определяет относительное влияние характеристического и символьного кодов друг на друга. Для того чтобы гарантировать, что данный характеристический код является доминантным, a выбирается равным 0,2. Входной вектор для каждого из животных состоит из 29 элементов и представляет собой объединение характеристического $\mathbf{x}_a$ и символьного $\mathbf{x}_s$ кодов:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_a \end{bmatrix} = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{x}_a \end{bmatrix}.$$

Наконец, каждый из векторов данных нормализуется к единичной длине. Примеры из таким образом созданного множества данных подаются на вход двумерной решетки нейронов размером 10×10, при этом синаптические веса нейронов настраиваются в соответствии с алгоритмом SOM (см. рис. 9.4). Обучение продолжалось в течение 2000 итераций, после чего карта признаков должна была достичь стабильного состояния. После этого самоорганизующейся сети подавались на вход векторы $\mathbf{x}=[\mathbf{x}_s,\mathbf{0}]^T$,

собака	.	.	лиса	.	.	кошка	.	.	орел
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	сова
.	.	.	.	.	.	тигр	.	.	.
волк	.	.	.	.	.	.	.	.	ястреб
.	.	.	лев	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	голубь
лошадь	.	.	.	.	.	.	курица	.	.
.	.	.	.	корова	.	.	.	.	гусь
зебра	.	.	.	.	.	.	утка	.	.

Рис. 9.17. Карта признаков, содержащая маркированные нейроны, имеющие самый сильный отклик на определенный входной пример

содержащие только символьные коды одного из животных, и определялся нейрон с самым сильным откликом. Эта процедура повторялась для всех 16 животных.

В процессе выполнения описанного алгоритма была получена карта, показанная на рис. 9.17. В ней маркированные нейроны дали самый сильный отклик на соответствующие примеры. Точками на карте обозначены нейроны с более слабыми откликами.

На рис. 9.18 показан результат моделирования для той же самоорганизующейся сети. Однако на этот раз каждый из нейронов сети был маркирован названием того животного, для которого его отклик был самым сильным. На этом рисунке видно, что карта признаков четко отслеживает "родственные взаимосвязи" между 16 различными животными. На нем можно выделить три обособленных кластера, один из которых представляет птиц, второй — мирных животных, а третий — хищников.

Карта признаков, показанная на рис. 9.18, называется *контекстной* (contextual) или *семантической* (semantic) [568], [888]. Эти карты напоминают вычислительные карты, формируемые в коре головного мозга (см. главу 2). Контекстные карты, получившиеся в результате использования алгоритма SOM, нашли свое применение в решении задач создания фонетических классов для текста, исследования Земли или дистанционного зондирования [568], а также в исследовании *данных и извлечении скрытых закономерностей* (data exploration & data mining) [569].

9.11. Резюме и обсуждение

По Кохонену [579] самоорганизующиеся карты представляют собой нейронные сети, построенные на одно- или двухмерной решетке нейронов для извлечения важных признаков, содержащихся во входном пространстве. Таким образом, можно получить структурное представление входных данных, используя в качестве прототипов векторы весов нейронов. Алгоритм SOM имеет нейробиологическую подоплеку. Он объединил в себе все основные механизмы, присущие самоорганизации: конкуренцию,

собака	собака	лиса	лиса	лиса	кошка	кошка	кошка	орел	орел
собака	собака	лиса	лиса	лиса	кошка	кошка	кошка	орел	орел
волк	волк	волк	лиса	кошка	тигр	тигр	тигр	сова	сова
волк	волк	лев	лев	лев	тигр	тигр	тигр	ястреб	ястреб
волк	волк	лев	лев	лев	тигр	тигр	тигр	ястреб	ястреб
волк	волк	лев	лев	лев	сова	голубь	ястреб	голубь	голубь
лошадь	лошадь	лев	лев	лев	голубь	курица	курица	голубь	голубь
лошадь	лошадь	зебра	корова	корова	корова	курица	курица	голубь	голубь
зебра	зебра	зебра	корова	корова	корова	курица	курица	утка	гусь
зебра	зебра	зебра	корова	корова	корова	утка	утка	утка	гусь

Рис. 9.18. Семантическая карта, полученная при использовании моделирования проводимости электродов. Эта карта разделена на три области, представляющие птиц, мирных животных и хищников

кооперацию и самоусиление (см. главу 8). Таким образом, его можно рассматривать как универсальный, несмотря на неполное развитие модели, описывающей возникновение явления коллективного порядка в сложных системах, которые начинаются с полного беспорядка.

Самоорганизующиеся карты можно также рассматривать как векторный квантователь, реализуя таким образом принципиальный подход к выводу правила коррекции, используемого для настройки векторов весов [691]. Этот последний подход еще раз подчеркивает роль функции окрестности, выступающей в роли функции плотности вероятности.

Однако следует особо отметить, что последний подход, основанный на использовании в (9.19) в качестве минимизируемой функции стоимости среднего распределения D_1, может быть оправдан только в том случае, если карта признаков уже достаточно хорошо упорядочена. В [285] показано, что динамика обучения самоорганизующейся карты на этапе упорядочивания адаптивного процесса (т.е. во время топологического упорядочивания изначально беспорядочной карты признаков) *не может* быть описана стохастическим градиентным спуском с *одной* функцией стоимости. Однако в случае одномерной решетки этот процесс может быть описан с использованием множества функций стоимости — по одной для каждого нейрона сети. При этом эти функции независимо минимизируются в соответствии с методом стохастического градиентного спуска.

В алгоритме SOM удивляет то, насколько он легко реализуем, и при этом довольно сложно анализировать его свойства, используя математический аппарат. Для его анализа некоторыми исследователями был предложен ряд мощных методов, однако их результаты имеют ограниченную область применения. В [214] был приведен обзор результатов, полученных в теоретических аспектах алгоритма SOM. В частности, были подчеркнуты недавние результаты, полученные в [304], [305], где утверждалось, что в случае одномерной решетки можно получить строгое доказательство *сходимости*

"почти наверняка" алгоритма SOM к единственному состоянию после прохождения им этапа самоорганизации. Этот важный результат был показан для довольно широкого класса функций окрестности. Однако то же самое нельзя утверждать для многомерных решеток.

Последним вопросом рассмотрения стал порядок. Так как самоорганизующиеся карты были навеяны идеями, полученными при исследовании карт коры головного мозга, вполне естественным выглядит то, что эта модель может описывать формирование этих самых карт мозга. Такое исследование было представлено в [283]. В этой работе показано, что самоорганизующаяся карта признаков способна объяснить формирование вычислительных карт в первичной *зрительной коре мозга* (primary visual cortex) макаки. Использовавшееся в этом исследовании входное пространство было пятимерным. Две размерности использовались для описания положения поля восприятия в пространстве сетчатки, а оставшиеся три — для *предпочтения ориентации* (orientation preference), *ориентационной избирательности* (orientation selectivity) и *окулярного доминирования* (ocular dominance). Поверхность коры мозга была поделена на небольшие участки, которые рассматривались как вычислительные элементы (т.е. искусственные нейроны) на двумерной квадратной решетке. При некоторых допущениях было показано, что обучение Хебба ведет к созданию пространственных моделей ориентации и окулярного доминирования, которые поразительно сходны с обнаруженными в мозге макаки.

Задачи

Алгоритм SOM

9.1. Пусть $g(y_j)$ — некоторая нелинейная функция отклика y_j, которая используется в алгоритме SOM (9.9). Что случится, если постоянное слагаемое в разложении функции $g(y_j)$ в ряд Тейлора не равно нулю.

9.2. Пусть $\pi(\mathbf{v})$ — гладкая функция шума в модели на рис. 9.6. Используя разложение Тейлора меры искажения (9.19), определите слагаемое искривления, возникающее в модели шума $\pi(\mathbf{v})$.

9.3. Иногда говорят, что алгоритм SOM *сохраняет* топологические связи, существующие во входном пространстве. Грубо говоря, это свойство можно гарантировать только для входного пространства с размерностью, меньшей или равной размерности нейронной решетки. Обсудите справедливость этого утверждения.

9.4. Говорят, что алгоритм SOM, основанный на конкурентном обучении, не обладает отказоустойчивостью. Это значит, что этот алгоритм толерантен к

ошибке, т.е. небольшое возмущение входного вектора способно вызвать резкий переход выхода от победившего нейрона к соседнему. Обсудите справедливость этих утверждений.

9.5. Рассмотрим пакетную версию алгоритма SOM, полученную представлением выражения (9.23) в его дискретной форме:

$$\mathbf{w}_j = \frac{\sum_i \pi_{j,i}\mathbf{x}_i}{\sum_i \pi_{j,i}}, \quad j = 1, 2, \ldots, l.$$

Покажите, что эта версия алгоритма SOM может быть выражена в форме, описанной в [186] (см. главу 5).

Квантизация векторов обучения

9.6. В этой задаче рассмотрим оптимизированную форму алгоритма квантизации векторов обучения (см. раздел 9.7) [568]. При этом требуется систематизировать эффект от коррекции векторов Вороного в разное время, чтобы получить равное влияние в конце периода обучения.

В первую очередь покажите, что уравнения (9.30) и (9.31) можно объединить в одно, имеющее следующий вид:

$$\mathbf{w}_c(n+1) = (1 - s_n\alpha_n)\mathbf{w}_c(n) + s_n\alpha_n\mathbf{x}(n),$$

где

$$s_n = \begin{cases} +1 & \text{при корректной классификации,} \\ -1 & \text{при некорректной классификации.} \end{cases} \tag{9.36}$$

Исходя из этого, покажите, что критерий оптимизации, описанный в постановке задачи, удовлетворяется, если

$$\alpha_n = (1 - s_n\alpha_n)\alpha_{n-1}, \tag{9.37}$$

а оптимальное значение константы обучения α_n равно

$$\alpha_n^{\text{opt}} = \frac{\alpha_{n-1}^{\text{opt}}}{1 + s_n\alpha_{n-1}^{\text{opt}}}.$$

9.7. Правила коррекции для обоих собственных фильтров максимума, о которых речь шла в главе 8, и для самоорганизующейся карты используют модификации постулата обучения Хебба. Сравните эти две модификации и найдите в них сходства и отличия.

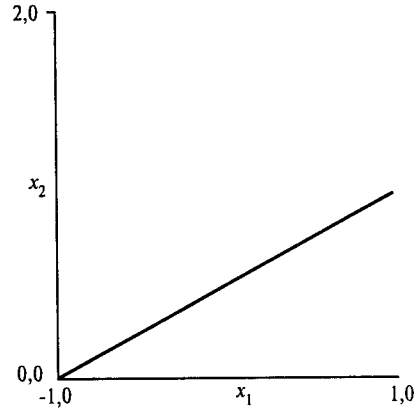


Рис. 9.19. Линейная плотность распределения входных данных

9.8. *Алгоритм с учетом "совести"* является модификацией алгоритма SOM, которая приводит к точному соответствию плотности распределения [254]. В этом алгоритме (табл. 9.4) каждый из нейронов отслеживает, сколько раз он победил в соревновании (т.е. сколько раз его вектор синаптических весов был ближе других нейронов к вектору входного сигнала в смысле Евклидова расстояния). Здесь следует сделать замечание, что если нейрон побеждает слишком часто, он "чувствует себя виновным" (feels guilty) и выводит себя из конкуренции.

Для исследования улучшения подобия плотности, вызывасмого алгоритмом с учетом "совести", рассмотрим одномерную решетку (т.е. линейный массив), состоящую из 20 нейронов, которые обучаются на входных примерах с линейной плотностью (рис. 9.19).

- С помощью компьютерного моделирования сравните соответствие плотности в результате применения алгоритмов SOM и алгоритма с учетом "совести". В алгоритме SOM параметр скорости обучения $\eta = 0,05$, а в алгоритме с учетом "совести" используются следующие параметры: $B = 0,0001$, $C = 1,0$, $\eta = 0,05$.
- Как часть эксперимента рассмотрите "точное" соответствие входной плотности.

Обсудите результаты компьютерного моделирования.

Компьютерные эксперименты

9.9. В этом эксперименте для исследования алгоритма SOM, примененного к одномерной решетке с двумерным входом, будем использовать компьютерное моделирование. Эта решетка состоит из 65 нейронов. Входы состоят из

ТАБЛИЦА 9.4. Алгоритм с учетом "совести" в сжатом виде

1. Находим вектор синаптических весов $\mathbf{w}_i$, ближайший к входному вектору $\mathbf{x}$:
$$\|\mathbf{x} - \mathbf{w}_i\| = \min_j \|\mathbf{x} - \mathbf{w}_j\|, \quad j = 1, 2, \ldots, N$$
2. Вычисляем общее количество моментов времени, p_j, в которые нейрон j победил в соревновании:
$$p_j^{\text{new}} = p_j^{\text{old}} + B(y_j - p_j^{\text{old}}), \text{ где } 0 < B \ll 1 \text{ и}$$
$$y_j = \begin{cases} 1, & \text{если нейрон } j \text{ является победившим,} \\ 0 - & \text{в противном случае.} \end{cases}$$
В начале алгоритма p_j инициализируется значением нуль.
3. Находим победивший нейрон, используя механизм conscience:
$$\|\mathbf{x} - \mathbf{w}_i\| = \min_j \left(\|\mathbf{x} - \mathbf{w}_j\| - b_j\right),$$
где b_j — слагаемое смещения, модифицирующее соревнование:
$$b_j = C\left(\frac{1}{N} - p_j\right),$$
где C — множитель смещения; N — общее количество нейронов в сети
4. Корректируем вектор синаптических весов победившего нейрона:
$$\mathbf{w}_i^{\text{new}} = \mathbf{w}_i^{\text{old}} + \eta(\mathbf{x} - \mathbf{w}_i^{\text{old}}),$$
где η — обычный параметр скорости обучения, используемый в алгоритме SOM

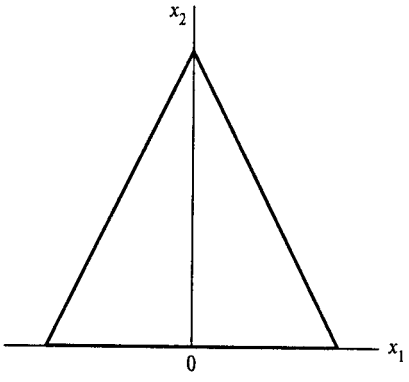


Рис. 9.20. Треугольная область распределения входных данных

случайных точек, равномерно распределенных в треугольной области, показанной на рис. 9.20. Вычислите карту, полученную алгоритмом SOM после 0, 20, 100, 1000, 10000 и 25000 итераций.

9.10. Рассмотрите двумерную решетку нейронов, обучаемую на основе входных сигналов с трехмерным распределением. Данная решетка имеет размер 10×10 нейронов.

Входной сигнал равномерно распределен в области, определенной следующим образом:

$$\{(0 < x_1 < 1), (0 < x_2 < 1), (0 < x_3 < 0.2)\}.$$

Для вычисления двумерной проекции входного пространства после 50, 1000 и 10000 итераций используйте алгоритм SOM.

Повторите вычисления для случая, когда вход равномерно распределен в параллелепипеде:

$$\{(0 < x_1 < 1), (0 < x_2 < 1), (0 < x_3 < 0.4)\}.$$

Повторите вычисления еще раз для входа, равномерно распределенного в кубе:

$$\{(0 < x_1 < 1), (0 < x_2 < 1), (0 < x_3 < 1)\}.$$

Обсудите результаты проведенного моделирования.

9.11. При использовании алгоритма SOM иногда возникает проблема топологического упорядочивания, называемая "скручиванием" (folded) карты. Эта проблема возникает в случаях, когда размер окрестности убывает слишком быстро. Создание скрученной карты можно рассматривать как одну из форм "локального минимума" процесса топологического упорядочивания.

Для того чтобы исследовать это явление, рассмотрим двумерную решетку размером 10×20 нейронов, обучаемую на основе двумерных входов, равномерно распределенных в квадрате $\{(-1 < x_1 < +1), (-1 < x_2 < +1)\}$. С помощью алгоритма SOM вычислите карту, сужая при этом окрестность победившего нейрона быстрее обычного. Чтобы явно увидеть ошибку упорядочивания, нужно повторить этот эксперимент несколько раз.

9.12. Свойство топологического упорядочивания алгоритма SOM можно использовать для формирования абстрактного двумерного представления входного пространства более высокого порядка. Для того чтобы исследовать эту форму представления, рассмотрим двумерную решетку размером 10×10 нейронов, обучаемую на основе входов, состоящих из четырех гауссовых множеств, $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{C}_3$ и $\mathbf{C}_4$. Все эти множества имеют единичную дисперсию, но разные центры: $(0, 0, \ldots, 0)$, $(4, 0, 0, \ldots, 0)$, $(4, 4, 0, \ldots, 0)$ и $(0, 4, 0, \ldots, 0)$. С помощью алгоритма SOM вычислите карту и пометьте каждый из нейронов этой карты именем того класса, который наиболее полно представлен окружающими его входными точками.

9.13. В табл. 9.5 представлен в сжатом виде *ренормализованный алгоритм SOM* (renormalized SOM algorithm). Краткое описание этого алгоритма см. в разделе 9.3. Сравните обычный алгоритм SOM с ренормализованным, обратив внимание на следующие вопросы.

1. Сложность кодирования в реализации алгоритма.
2. Время, затрачиваемое компьютером на обучение.

Проиллюстрируйте результаты сравнения этих двух алгоритмов, используя данные, взятые из равномерного распределения внутри квадрата, и следующие две конфигурации сети.

1. Одномерная решетка из 257 нейронов.
2. Одномерная решетка из 2049 нейронов.

В обоих случаях начните с исходного количества векторов кодирования, равного двум.

9.14. Рассмотрим пространственно-сигнальную диаграмму, показанную на рис. 9.21. Она соответствует *M-мерной амплитудно-импульсной модуляции* (M-ary pulse-amplitude modulation) для $M = 8$. Каждая точка сигнала представлена прямоугольным импульсом с соответствующей амплитудой:

$$p(t) = \pm\frac{7}{2}, \pm\frac{5}{2}, \pm\frac{3}{2}, \pm\frac{1}{2}, \quad 0 \le t \le T,$$

где T — интервал сигнала. На входе получателя к сигналу добавляется гауссов шум с нулевым средним, меняющий соотношение сигнал/шум (SNR). SNR определяется как отношение "средней" мощности переданного сигнала к мощности аддитивного шума.

а) Используя случайную двоичную последовательность на входе передатчика, сгенерируйте данные, полученные приемником для SNR=10, 20 и 30 дБ.

б) Для каждого из этих значений SNR сгенерируйте самоорганизующуюся карту. Для типичных значений можно использовать следующее.

Входной вектор, составленный из восьми элементов, полученных подвыборкой восьми элементов из одного интервала дискретизации сигнала. (Знаний о временных характеристиках сигнала не предполагается.)
Одномерная решетка, состоящая из 64 нейронов (т.е. восьмикратный размер входного вектора).

в) Нарисуйте карту признаков для каждого из трех SNR и таким образом продемонстрируйте свойство топологического самоупорядочивания алгоритма SOM.

Рис. 9.21. Пространственно-сигнальная диаграмма

ТАБЛИЦА 9.5. Алгоритм ренормализованного обучения (одномерная версия)

1. Инициализация	Устанавливаем количество векторов кодирования в некоторое малое число (для простоты можно использовать число 2 или некоторое большее значение, соответствующее сложности задачи). Устанавливаем их положение в случайно выбранные примеры обучения
2. Выбор входного вектора	Из множества примеров обучения случайным образом выбираем входной вектор
3. Кодирование входного вектора	Определяем "победивший" входной вектор (т.е. вектор синаптических весов нейрона-победителя). Для этого можно использовать правило "ближайшего соседа" или "минимального искажения"
4. Корректировка кодовой книги	Производим коррекцию победившего нейрона и его "топологических соседей". При этом можно оставить параметр интенсивности обучения на некотором фиксированном значении (например, на 0,125) и использовать для победившего нейрона коэффициент η, а для его соседей — $\eta/2$
5. Разбивка кодовой книги	Продолжаем корректировку кодовой книги (шаг 4), каждый раз используя новый входной вектор, случайно выбираемый из множества примеров обучения, пока количество этих корректировок в 10–30 раз не превзойдет количество векторов кодирования. После этого кодовая книга разбивается. Это можно выполнить с помощью строки Пеано (Peano string) имеющихся векторов кодирования и такой интерполяцией их положений, которая обеспечивает самое точное приближение строки Пеано. Для этого можно поместить дополнительный вектор кодирования между двумя уже существующими
6. Завершение обучения	Корректировка и разбивка кодовой книги продолжается до тех пор, пока общее количество векторов кодирования не достигнет некоторого наперед заданного значения (например, 100). После этого обучение прекращается

Примечание. Разбивка кодовой книги после каждой эпохи приблизительно удваивает количество векторов кодирования, поэтому для этого процесса требуется не так уж много эпох.

Модели на основе теории информации

10.1. Введение

В своей классической работе Клод Шеннон (Claude Shennon) изложил основы *теории информации* (information theory) [970]. Эта работа[1], а также дальнейшее развитие теории информации другими исследователями были ответом на потребность специалистов в области электроники в создании одновременно *эффективных* и *надежных* систем. В свете практических применений теория информации в том виде, в котором она существует сегодня, является математической теорией, проникающей в самую сущность *процесса взаимодействия* (communication process). Эта теория открыла простор для изучения таких фундаментальных вопросов, как эффективность представления информации и ограничения, налагаемые на надежность передачи информации по каналам связи. Более того, эта теория охватывает массу мощнейших теорем, вычисляющих идеальные границы для оптимального представления и передачи информационных сигналов. Эти ограничения являются чрезвычайно важными, так как они описывают критерии улучшения архитектуры систем обработки информации.

Главной целью настоящей главы является рассмотрение *информационно-теоретических моделей* (information-theoretic model), которые приводят к самоорганизации. В этом контексте моделью, которая имеет особое значение, является *принцип максимума взаимной информации*[2] (maximum mutual information principle) Линскера [653], [654]. Этот принцип гласит, что синаптические связи многослойной нейронной сети организуются таким образом, чтобы *при определенных условиях мак-*

---

[1] Детальное изложение теории информации содержится в [221] и [377]. Также можно обратиться к сборнику работ [1000], содержащему среди прочих и классическую работу Шеннона [970]. Работа Шеннона также воспроизведена с незначительными измеиениями в [971] и [1001].

Краткое изложение основных принципов теории информации, касающихся нейронных сетей, содержится в [77]. Трактовка теории информации с биологической точки зреиия содержится в [1173].

[2] Не следует путать принцип максимума взаимной информации Лиискера для самоорганизующихся систем с правнлом сохранеиия контекста информации (information-context preservation rule) в принятии решений практического метода, рассмотрениого в главе 7.

симизировать объем информации, которая сохраняется при преобразовании сигнала на каждой из стадий обработки сигнала в сети. Эта идея, предложенная теорией информации для описания обработки сигнала, не является новой[3]. Достаточно вспомнить раннюю работу, в которой для систем обработки информации предложена следующая информационно-теоретическая функция [83].

Главной функцией систем обработки сигнала является раскрытие некоторой избыточности стимулирования для описания или кодировки информации в форме, более экономной, чем та, в которой она воспринимается рецепторами.

Главной идеей работы [83] является признание того, что кодирование данных с целью уменьшения избыточности связано с идентификацией их специфичных свойств. Эта идея связана с особой точкой зрения на мозг, приведенной в [230], где модель внешнего мира строилась как набор закономерностей и ограничений.

Структура главы

Эта глава состоит из двух основных частей. В первой части (разделы 10.2–10.5) излагаются основы теории информации. В разделе 10.2 описывается концепция энтропии как количественной меры информации, что естественным образом приведет к принципу максимума энтропии, описываемому в разделе 10.3. Далее, в разделе 10.4, мы обсудим концепцию взаимной информации, затем в разделе 10.5 последует описание дивергенции Кулбека–Лейблера.

Вторая часть этой главы (разделы 10.6–10.14) посвящена информационно-теоретическим моделям самоорганизующихся систем. В разделе 10.6 взаимная информация выводится в ранг оптимизируемых целевых функций. В разделе 10.7 описывается принцип максимума взаимной информации, за чем в разделе 10.8 следует рассмотрение взаимосвязи между этим принципом и уменьшением избыточности. В разделах 10.9, 10.10 рассматриваются два варианта принципа взаимной информации, которые применимы к разным задачам обработки изображений. В разделах 10.11–10.14 описываются три различных метода решения задачи *слепого разделения источников сигнала* (blind source separation problem).

Как всегда, глава завершается некоторыми заключительными замечаниями.

10.2. Энтропия

Следуя терминологии, принятой в теории вероятности, для обозначения *случайных переменных* будем использовать прописные буквы, при этом одноименными строчными буквами обозначим их *значения*.

---

[3] Обзор литературы, посвященной связи теории информации с восприятнем (perception), содержится в [77] и [650].

Рассмотрим случайную переменную X, каждая из реализаций которой может рассматриваться как *сообщение* (message). Грубо говоря, если случайная переменная X является непрерывной на всей своей области значений, то она несет в себе бесконечный объем информации. Однако с биологической и физической точки зрения бессмысленно рассуждать в терминах измерения амплитуды с бесконечной точностью. Таким образом, значения переменной X могут быть равномерно *дискретизированы* на конечное число уровней. Следовательно, X можно рассматривать как *дискретную* случайную переменную, моделируемую следующим образом:

$$X = \{x_k | k = 0, \pm 1, \ldots, \pm K\}, \tag{10.1}$$

где x_k — дискретное число; $(2K + 1)$ — общее количество дискретных уровней. Различие δx между дискретными уровнями предполагается достаточно малым, чтобы обеспечить адекватное представление рассматриваемой переменной. Можно, естественно, принять за δx число нуль и перейти к множеству мощности континуум. В этом случае мы получим непрерывную случайную переменную, для которой суммирование нужно заменить интегрированием (что будет показано далее).

Чтобы завершить описание модели, примем следующее допущение: пусть $X = x_k$ с *вероятностью*

$$p_k = P(X = x_k) \tag{10.2}$$

при условии

$$0 \leq p_k \leq 1 \text{ и } \sum_{k=-K}^{K} p_k = 1. \tag{10.3}$$

Предположим, что событие $X = x_k$ наступает с вероятностью $p_k = 1$. Отсюда следует, что $p_i = 0$ для всех $i \neq k$. В этом случае не будет неожиданности, а следовательно и новой информации, в наступлении события $X = x_k$, так как мы знаем наверняка, что это событие должно произойти. Если же, с другой стороны, различные уровни могут достигаться с различной вероятностью и если эта вероятность p_k достаточно мала, то наступление события $X = x_k$, а не $X = x_i$ будет нести в себе больше информации. Таким образом, слова "неопределенность", "неожиданность" и "информация" тесно взаимосвязаны. В наступлении события $X = x_k$ есть большая доля неопределенности. В том, что это событие действительно произошло, есть элемент неожиданности или сюрприза. После наступления события $X = x_k$ наблюдается увеличение объема информации. Эти три величины по сути являются одним и тем же. Более того, объем информации является величиной, *обратной* к вероятности возникновения события.

Определим объем информации, собранный с наступлением события $X = x_k$ с вероятностью p_k, как следующую логарифмическую функцию:

$$I(x_k) = \log\left(\frac{1}{p_k}\right) = -\log p_k, \tag{10.4}$$

где основание логарифма — произвольно. Если используется натуральный логарифм, единица информации называется *нат* (nat), а если логарифм по основанию 2 — то *бит* (bit). В любом случае определение информации (10.4) обладает рядом следующих свойств.

1. $$I(x_k) = 0, \text{ если } p_k = 1. \tag{10.5}$$

Если мы абсолютно уверены в наступлении определенного события, то его возникновение не несет в себе информации.

2. $$I(x_k) \geq 0, \text{ если } 0 \leq p_k \leq 1. \tag{10.6}$$

Это значит, что возникновение события $X = x_k$ либо несет в себе информацию, либо нет, но никогда не приводит к потере информации.

3. $$I(x_k) > I(x_i), \text{ если } p_k < p_i. \tag{10.7}$$

Чем меньше вероятность события, тем больше информации несет в себе его наступление.

Объем информации $I(x_k)$ является дискретной случайной переменной с вероятностью p_k. Среднее значение $I(x_k)$ на полном диапазоне $(2K+1)$ дискретных значений вычисляется следующим образом:

$$H(X) = E[I(x_k)] = \sum_{k=-K}^{K} p_k I(x_k) = -\sum_{k=-K}^{K} p_k \log p_k. \tag{10.8}$$

Величина $H(X)$ называется *энтропией* (entropy) случайной переменной X, принимающей конечное множество дискретных значений. Это название еще раз подчеркивает аналогию между определением (10.8) и определением энтропии в статистической термодинамике[4]. Энтропия $H(X)$ является *мерой среднего объема информации*,

[4] Термин "энтропия" в теории информации получил свое название по аналогии с энтропией в термодинамике, описываемой следующей формулой (см. главу 11): $H = -k_B \sum_{\alpha} p_\alpha \log p_\alpha$, где k_B — константа Больцмана; p_α — вероятность того, что система находится в состоянии α. За исключением множителя k_B энтропия H в термодинамике имеет точно такую же математическую формулу, что и формула энтропии (10.8).

которую содержит в себе сообщение. Однако обратите внимание на то, что X в обозначении $H(X)$ не является аргументом функции, а представляет собой метку случайной переменной. Также обратите внимание на то, что в определении (10.8) значением произведения $0 \cdot \log(0)$ считается нуль.

Энтропия $H(X)$ может принимать следующие значения:

$$0 \leq H(X) \leq \log(2K+1), \tag{10.9}$$

где $(2K+1)$ — общее количество дискретных уровней. Более того, можно сформулировать следующие утверждения.

1. $H(X) = 0$ тогда и только тогда, когда $p_k = 1$ для некоторого k, а для всех остальных уровней вероятность равна нулю. Эта нижняя граница энтропии соответствует отсутствию *неопределенности*.
2. $H(K) = \log_2(2K+1)$ тогда и только тогда, когда $p_k = 1/(2K+1)$ для всех k (т.е. если все дискретные значения равновероятны). Эта верхняя граница энтропии соответствует *максимальной неопределенности*.

Доказательство второго свойства вытекает непосредственно из следующей леммы [377].

Для любых двух распределений $\{p_k\}$ и $\{q_k\}$ дискретной случайной переменной X неравенство

$$\sum_k p_k \log\left(\frac{p_k}{q_k}\right) \geq 0 \tag{10.10}$$

обращается в равенство тогда и только тогда, когда $q_k = p_k$ для всех k.

Величина, используемая в приведенной лемме, имеет такое фундаментальное значение, что мы остановимся, чтобы привести ее к виду, более удобному для изучения стохастических систем. Пусть $p_X(x)$ и $q_X(x)$ — вероятности того, что случайная переменная X находится в состоянии x при двух различных условиях. *Относительная энтропия* (relative entropy) или *дивергенция Кулбека–Лейблера* (Kullback-Leibler divergence) между двумя *вероятностными функциями массы* (probability mass function) определяется следующим образом [221], [377], [605]:

$$D_{p||q} = \sum_{x \in \mathbf{X}} p_X(x) \log\left(\frac{p_X(x)}{q_X(x)}\right), \tag{10.11}$$

где суммирование проводится по всем возможным состояниям системы (т.е. по алфавиту $\mathbf{X}$ дискретной случайной переменной X). Функция массы вероятности $q_X(x)$ выступает в роли *меры ссылки* (reference measure).

Дифференциальная энтропия непрерывной случайной переменной

В рассмотрении информационно-теоретических концепций до сих пор участвовали только дискретные случайные переменные. Некоторые из этих концепций можно распространить и на непрерывные случайные переменные.

Рассмотрим случайную переменную X, имеющую *функцию плотности вероятности* $f_X(x)$. По аналогии с энтропией дискретной случайной переменной введем следующее определение:

$$h(X) = -\int_{-\infty}^{\infty} f_X(x) \log f_X(x) dx = -E[\log f_X(x)]. \tag{10.12}$$

Величина $h(X)$ называется *дифференциальной энтропией* (differential entropy) случайной переменной X, что отличает ее от обычной или *абсолютной энтропии* (absolute entropy). Нельзя не признать тот факт, что несмотря на то, что величина $h(X)$ является полезной с математической точки зрения, она *не* имеет никакого значения в смысле меры случайности переменной X.

Теперь обоснуем использование формулы (10.12). Начнем с рассмотрения непрерывной случайной переменной X как предельной формы дискретной случайной переменной $x_k = k\delta x$, где $k = 0, \pm 1, \pm 2, \ldots$, а δx стремится к нулю. По определению непрерывная случайная переменная X принимает значения в интервале $[x_k, x_k + \delta x]$ с вероятностью $f_X(x_k)\delta x$. Исходя из этого, устремляя δx к нулю, обычную энтропию непрерывной случайной переменной X можно выразить через следующий предел:

$$\begin{aligned}
H(X) &= -\lim_{\delta x \to 0} \sum_{k=-\infty}^{\infty} f_X(x_k)\delta x \log(f_X(x_k)\delta x) = \\
&= -\lim_{\delta x \to 0} \left[\sum_{k=-\infty}^{\infty} f_X(x_k) \log(f_X(x_k)\delta x) + \log \delta x \sum_{k=-\infty}^{\infty} f_X(x_k)\delta x \right] = \\
&= -\int_{-\infty}^{\infty} f_X(x) \log f_X(x) dx - \lim_{\delta x \to 0} \log \delta x \int_{-\infty}^{\infty} f_X(x) dx = \\
&= h(X) - \lim_{\delta x \to 0} \log \delta x,
\end{aligned} \tag{10.13}$$

где в последней строке использовалось выражение (10.12) и тот факт, что общая площадь области под кривой функции плотности вероятности $f_X(x)$ равна единице. В пределе, по мере достижения величиной δx нуля, значение $-\log \delta x$ достигает бесконечности. Интуитивно это понятно, так как непрерывная случайная переменная принимает все значения из открытого интервала $(-\infty, \infty)$, а неопределенность, ассоциированная с этой переменной, имеет порядок бесконечности. Мы избегаем проблем, связанных со слагаемым $-\log \delta x$, принимая $h(X)$ за дифференциальную энтропию, а слагаемое $-\log \delta x$ за ссылку. Более того, так как информация, обрабатываемая стохастической системой, представляет собой разницу между двумя слагаемыми эн-

тропии, которые имеют общую ссылку, информация будет такой же, как и разность между соответствующими слагаемыми дифференциальной энтропии. Таким образом, мы обосновали использование слагаемого $h(X)$, определенного согласно (10.13), в качестве дифференциальной энтропии непрерывной случайной переменной X.

При использовании вектора $\mathbf{X}$, состоящего из n случайных переменных $X_1, X_2, \ldots, X_n$, дифференциальная энтропия определяется как n-й интеграл

$$h(\mathbf{X}) = -\int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = -E[\log f_{\mathbf{X}}(\mathbf{x})], \tag{10.14}$$

где $f_{\mathbf{X}}(\mathbf{x})$ — функция плотности совместной вероятности $\mathbf{X}$.

Пример 10.1

Равномерное распределение

Рассмотрим случайную переменную X, равномерно распределенную в интервале [0,1]:

$$f_X(x) = \begin{cases} 1, & 0 \le x \le 1, \\ 0 - \text{в противном случае.} \end{cases}$$

Применяя выражение (10.12), получаем, что дифференциальная энтропия равна нулю:

$$h(X) = -\int_{-\infty}^{\infty} 1 \cdot \log 1 dx = -\int_{-\infty}^{\infty} 1 \cdot 0 dx = 0.$$

■

Свойства дифференциальной энтропии

Из определения дифференциальной энтропии $h(X)$ (10.12) видно, что при перемещении ее значение не меняется, т.е.

$$h(X + c) = h(X), \tag{10.15}$$

где c — константа.

Еще одно важное свойство дифференциальной энтропии описывается следующим образом:

$$h(aX) = h(X) + \log |a|, \tag{10.16}$$

где a — масштабирующий множитель. Для того чтобы доказать это свойство, вспомним, что площадь области под кривой функции плотности вероятности равна единице, и тогда:

$$f_Y(y) = \frac{1}{|a|} f_Y\left(\frac{y}{a}\right). \tag{10.17}$$

Затем, используя формулу (10.12), можно записать:

$$h(Y) = -E[\log f_Y(y)] = -E\left[\log\left(\frac{1}{|a|}f_Y\left(\frac{y}{a}\right)\right)\right] = -E\left[\log f_Y\left(\frac{y}{a}\right)\right] + \log|a|\,.$$

Подставляя вместо Y произведение aX, получим:

$$h(aX) = -\int_{-\infty}^{\infty} f_X(x)\log f_X(x)dx + \log|a|\,,$$

из чего непосредственно вытекает свойство (10.16).

Свойство (10.16) применимо к скалярной случайной переменной. Его можно обобщить для случая произведения случайного вектора $\mathbf{X}$ и матрицы $\mathbf{A}$:

$$h(\mathbf{AX}) = h(\mathbf{X}) + \log|\det(\mathbf{A})|, \tag{10.18}$$

где $\det(\mathbf{A})$ — определитель матрицы $\mathbf{A}$.

10.3. Принцип максимума энтропии

Предположим, что существует некоторая стохастическая система с множеством известных состояний, но с неизвестными вероятностями, и что некоторым образом происходит процесс обучения ограничениям распределения вероятности этих состояний. Этими ограничениями могут быть некоторые средние по множеству значения или границы значений. Задача состоит в выборе такой модели вероятности, которая будет оптимальной в некотором смысле при наличии *априорных знаний* о модели. Обычно оказывается, что этим условиям удовлетворяет бесконечное множество моделей. Какую же модель из них выбрать?

Ответ на этот фундаментальный вопрос можно получить путем применения *принципа максимальной энтропии*[5] (maximum entropy principle) [512]. Этот принцип звучит следующим образом [511], [512].

---

[5] В [982] было доказано, что принцип максимальной энтропии корректен в следующем смысле.

Для данных знаний в форме ограничений существует только одно распределение, удовлетворяющее этим ограничениям, которое можно выбрать с помощью процедуры, удовлетворяющей "аксиомам согласованности" (consistency axioms). Это уникальное распределение определяется максимизацией энтропии.

Ниже приведены эти аксиомы согласованности.

1. Уникальность. Результат должен быть уникальным.
2. Инвариантность. Выбор системы координат не должен влиять на результат.
3. Независимость системы. Не должно иметь значения, будет ли независимая информация о независимых системах браться в расчет как в терминах различных плотностей раздельно, так и вместе, в терминах совместной плотности.
4. Независимость подмножеств. Не должно иметь значения, будет ли независимое множество состояний системы рассматриваться в терминах условной плотности или полной плотности системы.

В [982] было показано, что относительная энтропия, или дивергенция Кулбека–Лейблера, также удовлетворяет аксиомам согласованности.

Если выводы основываются на неполной информации, она должна выбираться из распределения вероятности, максимизирующего энтропию при заданных ограничениях на распределение.

В результате понятие энтропии определяет некоторый род меры пространства распределений вероятности, при которой распределениям с высокой энтропией отдается предпочтение перед остальными.

Исходя из этого утверждения, становится совершенно очевидным, что задача максимизации энтропии является задачей условной оптимизации. Для иллюстрации решения этой задачи рассмотрим максимизацию дифференциальной энтропии

$$h(X) = -\int_{-\infty}^{\infty} f_X(x) \log f_X(x) dx$$

на всех функциях распределения вероятности $f_X(x)$ случайной переменной X при соблюдении следующих условий.

1. $f_X(x) \geq 0$, причем равенство достигается независимо от x.

2. $$\int_{-\infty}^{\infty} f_X(x) dx = 1.$$

3. $$\int_{-\infty}^{\infty} f_X(x) g_i(x) dx = \alpha_i, \; i = 1, 2, \ldots, m,$$

где $g_i(x)$ — некоторая функция от x.

Условия 1 и 2 являются фундаментальными свойствами функции плотности вероятности. Условие 3 определяет моменты X, которые зависят от способа определения функции $g_i(x)$. В результате условие 3 суммирует все априорные знания о случайной переменной X. Для того чтобы решить эту задачу условной оптимизации, будем использовать *метод множителей Лагранжа*[6]. Сначала составим целевую функцию:

$$J(f) = \int_{-\infty}^{\infty} \left[-f_X(x) \log f_X(x) + \lambda_0 f_X(x) + \sum_{i=1}^{m} \lambda_i g_i(x) f_X(x) \right] dx, \qquad (10.19)$$

где $\lambda_1, \lambda_2, \ldots, \lambda_m$ — *множители Лагранжа* (Lagrange multipliers). Дифференцируя интеграл по $f_X(x)$ и приравнивая результат к нулю, получим:

$$-1 - \log f_X(x) + \lambda_0 + \sum_{i=1}^{m} \lambda_i g_i(x) = 0.$$

[6] Метод множителей Лагранжа рассматривается в [263].

Решая это уравнение относительно неизвестной функции $f_X(x)$, получим:

$$f_X(x) = \exp\left(-1 + \lambda_0 + \sum_{i=1}^{m} \lambda_i g_i(x)\right). \tag{10.20}$$

Множители Лагранжа в (10.20) выбираются в соответствии с условиями 2 и 3. Равенство (10.20) определяет распределение с максимальной энтропией как решение задачи.

Пример 10.2

Одномерное гауссово распределение

Предположим, известны следующие характеристики случайной переменной X: ее среднее значение μ и дисперсия σ^2. По определению имеем:

$$\int_{-\infty}^{\infty} (x-\mu)^2 f_X(x)dx = \sigma^2 = \text{const}.$$

Сравнивая это равенство с условием 3, видим, что

$$g_1(x) = (x-\mu)^2,$$
$$\alpha^1 = \sigma^2.$$

Используя равенство (10.20), получим:

$$f_X(x) = \exp[-1 + \lambda_0 + \lambda_1(x-\mu)^2].$$

Обратите внимание, что λ_1 будет отрицательным, если интегралы от $f_X(x)$ и $(x-\sigma)^2 f_X(x)$ по x сходятся. Подставляя это равенство в условия 2 и 3, а затем разрешая результат относительно λ_0 и λ_1, получим:

$$\lambda_0 = 1 - \log(2\pi\sigma^2)$$

и

$$\lambda_1 = -\frac{1}{2\sigma^2}.$$

Таким образом, искомая форма функции $f_X(x)$ имеет следующий вид:

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{10.21}$$

в чем легко можно узнать *плотность вероятности гауссовой случайной переменной* X со средним значением μ и с дисперсией σ^2. Максимальное значение дифференциальной энтропии такой случайной переменной выглядит так:

$$h(X) = \frac{1}{2}[1 + \log(2\pi\sigma^2)]. \tag{10.22}$$

На основании результатов этого примера можно сделать следующие выводы.

1. *Для данной дисперсии* σ^2 *гауссова случайная переменная имеет наибольшую дифференциальную энтропию среди любых случайных переменных.* Это значит, что если X является гауссовой случайной переменной, а Y — любая другая случайная переменная с тем же средним значением и дисперсией, то для всех Y выполняется

$$h(X) \geq h(Y),$$

причем равенство достигается только при совпадении распределений X и Y.

2. *Энтропия гауссовой случайной переменной X однозначно определяется ее дисперсией* (т.е. не зависит от ее среднего значения). ■

Пример 10.3

Многомерное гауссово распределенне

В этом примере мы на основании результатов примера 10.2 оценим дифференциальную энтропию многомерного гауссова распределения. Так как энтропия гауссовой случайной переменной X не зависит от среднего значения, можно вполне обоснованно упростить рассмотрение, приняв среднее значение случайного вектора $\mathbf{X}$ размерности $m \times 1$ за нуль. Пусть статистика второго порядка $\mathbf{X}$ описывается матрицей ковариации $\boldsymbol{\Sigma}$, определенной как среднее значение произведения вектора $\mathbf{X}$ самого на себя. Функция плотности совместной вероятности случайного вектора $\mathbf{X}$ задается в следующем виде:

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{m/2}(\det(\boldsymbol{\Sigma}))^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}\right), \tag{10.23}$$

где $\det(\sigma)$ — определитель матрицы σ. Равенство (10.14) определяет дифференциальную энтропию $\mathbf{X}$. Исходя из этого, подставим (10.23) в (10.14) и получим:

$$h(\mathbf{X}) = \frac{1}{2}[m + m\log(2\pi) + \log|\det(\boldsymbol{\Sigma})|]. \tag{10.24}$$

Заметим, что равенство (10.22) является всего лишь частным случаем более общего равенства (10.24). В свете принципа максимума энтропии можно утверждать, что для данной матрицы ковариации $\boldsymbol{\Sigma}$ многомерное гауссово распределение (10.23) имеет наибольшую дифференциальную энтропию среди случайных векторов с нулевым средним значением. Этот максимум энтропии имеет значение, определяемое формулой (10.24). ■

10.4. Взаимная информация

При создании самоорганизующихся систем главной целью была разработка такого алгоритма, который способен обучаться отображению входа на выход на основе только входного сигнала. В этом контексте понятие взаимной информации имеет особое значение из-за своих отдельных свойств. Для того чтобы подготовить почву для дискуссии, рассмотрим стохастическую систему с входом X и выходом Y. X и Y могут принимать только *дискретные* значения — x и y соответственно. Энтропия $H(X)$ является мерой изначальной неопределенности X. Как можно измерить неопределенность X на основе наблюдения Y? Чтобы ответить на этот вопрос, введем понятие *условной энтропии* (conditional entropy) X и Y [221], [377]:

$$H(X|Y) = H(X,Y) - H(Y). \tag{10.25}$$

Условная энтропия имеет следующее свойство:

$$0 \leq H(X|Y) \leq H(X). \tag{10.26}$$

Условная энтропия $H(X|Y)$ представляет собой *уровень оставшейся неопределенности относительно X после того, как было получено наблюдение Y*. Другая величина в равенстве (10.25): $H(X,Y)$ является *совместной энтропией* (joint entropy), определяемой следующим образом:

$$H(X,Y) = -\sum_{x\in\mathbf{X}}\sum_{y\in\mathbf{Y}} p(x,y)\log p(x,y),$$

где $p(x,y)$ — *функция массы совместной вероятности* дискретных случайных переменных X и Y, а $\mathbf{X}$ и $\mathbf{Y}$ — их соответствующие алфавиты.

Так как энтропия $H(X)$ представляет неопределенность относительно входа системы перед наблюдением выхода системы, а условная энтропия $H(X|Y)$ — ту же неопределенность *после* наблюдения выхода, то разность $H(X) - H(X|Y)$ будет определять ту часть неопределенности, которая была разрешена наблюдением выхода системы. Эта величина называется *взаимной информацией* (mutual information) между случайными переменными X и Y. Обозначив эту величину как $I(X;Y)$, можно записать[7]:

$$I(X;Y) = H(X) - H(X|Y) = \sum_{x\in\mathbf{X}}\sum_{y\in\mathbf{Y}} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right). \qquad (10.27)$$

Энтропия является частным случаем взаимной информации, так как

$$H(X) = I(X;X).$$

Взаимная информация $I(X;Y)$ между двумя случайными переменными X и Y обладает следующими свойствами [221], [377].

1. *Взаимная информация между X и Y является симметричной*, т.е.

$$I(Y;X) = I(X;Y),$$

где взаимная информация $I(Y;X)$ является мерой неопределенности выхода Y, разрешенной при снятии данных с входа X; а взаимная информация $I(X;Y)$ — мерой неопределенности входа системы, разрешенной снятием наблюдения с выхода системы.

---

[7] Величина $I(X;Y)$ изначально называлась *скоростью передачи информации* (rate of information transmission) [970]. Однако сегодня она получила название взаимной информации между случайными переменными X и Y.

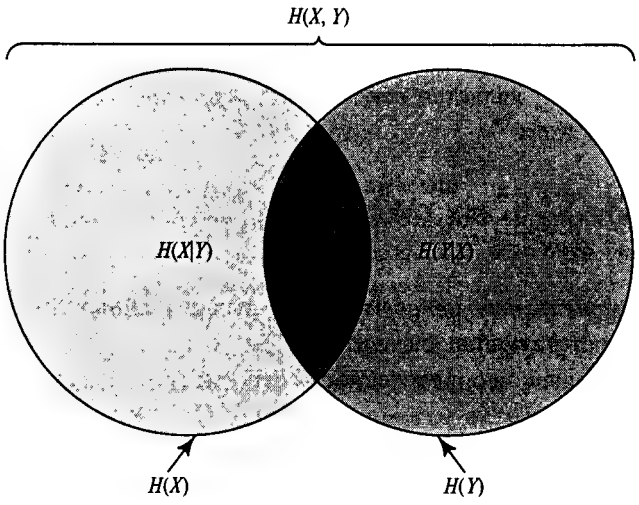


Рис. 10.1. Взаимосвязь информации $I(X|Y)$ с энтропиями $H(X)$ и $H(Y)$

2. *Взаимная информация между X и Y не может быть отрицательной*:

$$I(X;Y) \geq 0.$$

Это свойство взаимной информации означает, что при снятии наблюдения с выхода системы не может произойти потеря информации. Более того, взаимная информация равна нулю тогда и только тогда, когда вход и выход системы являются статистически независимыми.

3. *Взаимная информация между X и Y может быть выражена в терминах энтропии выхода Y следующим образом*:

$$I(X;Y) = H(Y) - H(Y|X), \tag{10.28}$$

где $H(Y|X)$ — условная энтропия. В правой части равенства (10.28) — усреднение по множеству информации, переданной выходом системы Y, за вычетом средней по множеству информации, учитывающей знание о входе X. Последняя величина, $H(Y|X)$, несет информацию о *помехах обработки* (processing noise), а не о самом входе системы X.

На рис. 10.1 представлена визуальная интерпретация равенств (10.27) и (10.28). Энтропия входа системы X представлена левым кругом, а энтропия выхода Y — правым. Взаимная информация между X и Y представлена пересечением этих двух кругов.

Взаимная информация непрерывных случайных переменных

Теперь рассмотрим пару непрерывных случайных переменных X и Y. По аналогии с формулой (10.27) *взаимную информацию* между этими переменными можно определить следующим образом:

$$I(X;Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{X,Y}(x,y)\log\left(\frac{f_X(x|y)}{f_X(x)}\right)dxdy, \tag{10.29}$$

где $f_{X,Y}(x,y)$ — функция плотности совместной вероятности X и Y; $f_X(x|y)$ — функция плотности условной вероятности X при $Y = y$. Обратите внимание, что

$$f_{X,Y}(x,y) = f_X(x|y)f_Y(y).$$

Исходя из этого, (10.29) можно переписать в следующем виде:

$$I(X;Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{X,Y}(x,y)\log\left(\frac{f_{X,Y}(x,y)}{f_X(x)f_Y(y)}\right)dxdy.$$

Также, по аналогии с предыдущим обсуждением дискретных случайных переменных, взаимная информация $I(X;Y)$ между непрерывными случайными переменными X и Y обладает следующими свойствами:

$$I(X;Y) = h(X) - h(X|Y) = h(Y) - h(Y|X) = h(X) + h(Y) - h(X,Y), \tag{10.30}$$

$$I(Y;X) = I(X;Y), \tag{10.31}$$

$$I(X;Y) \geq 0. \tag{10.32}$$

Параметр $h(X)$ является дифференциальной энтропией X. Аналогично можно сказать и о параметре $h(Y)$. Параметр $h(X|Y)$ является *условной дифференциальной энтропией* (conditional differential entropy) X для данного Y и определяется следующим двойным интегралом:

$$h(X|Y) = -\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{X,Y}(x,y)\log f_X(x|y)dxdy. \tag{10.33}$$

Параметр $h(Y|X)$ является условной дифференциальной энтропией Y для данного X. Он определяется аналогично параметру $h(X|Y)$. Параметр $h(X,Y)$ является совместной дифференциальной энтропией X и Y.

Обратите внимание, что в неравенстве (10.32) строгое равенство соблюдается тогда и только тогда, когда случайные переменные X и Y являются *статистически независимыми*. Если выполняется это условие, функция плотности совместной веро-

ятности X и Y может быть разложена на множители:

$$f_{X,Y}(x,y) = f_X(x)f_Y(y), \tag{10.34}$$

где $f_X(x)$ и $f_Y(y)$ — *граничные* (marginal) функции плотности вероятности X и Y соответственно. Эквивалентно можно записать:

$$f_X(x|y) = f_X(x).$$

Эта формула означает, что знание о выходе Y никак не влияет на распределение входного сигнала X. Применяя это условие к (10.29), сведем взаимную информацию $I(X;Y)$ между X и Y к нулю.

Определение взаимной информации $I(X;Y)$, представленное формулой (10.29), применимо к скалярным случайным переменным X и Y. Это определение можно естественным образом обобщить на случайные векторы $\mathbf{X}$ и $\mathbf{Y}$. Взаимная информация I($\mathbf{X}$,$\mathbf{Y}$) определяется как *многократный* (multifold) интеграл:

$$I(\mathbf{X};\mathbf{Y}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) \log\left(\frac{f_{\mathbf{X}}(\mathbf{x}|\mathbf{y})}{f_{\mathbf{X}}(\mathbf{x})}\right) d\mathbf{x}d\mathbf{y}. \tag{10.35}$$

Взаимная информация $I(\mathbf{X};\mathbf{Y})$ имеет те же свойства, которые представлены формулами (10.30)–(10.32) для скалярных случайных переменных.

10.5. Дивергенция Кулбека–Лейблера

Формула (10.11) определяет понятие дивергенции Кулбека–Лейблера для дискретных случайных переменных. Это определение можно расширить для более общего случая — непрерывных случайных векторов. Пусть $f_{\mathbf{X}}(\mathbf{x})$ и $g_{\mathbf{X}}(\mathbf{x})$ — две различные функции плотности вероятности случайного вектора $\mathbf{X}$ размерности $m \times 1$. В свете формулы (10.11) можно определить *дивергенцию Кулбека–Лейблера* между $f_{\mathbf{X}}(\mathbf{x})$ и $g_{\mathbf{X}}(\mathbf{x})$ следующим образом [605], [982]:

$$D_{f_{\mathbf{X}}||g_{\mathbf{X}}} = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{f_{\mathbf{X}}(\mathbf{x})}{g_{\mathbf{X}}(\mathbf{x})}\right) d\mathbf{x}. \tag{10.36}$$

Дивергенция Кулбека–Лейблера обладает несколькими уникальными свойствами.

1. Она не может быть отрицательной. В частном случае, когда $f_{\mathbf{X}}(\mathbf{x})$=$g_{\mathbf{X}}(\mathbf{x})$, т.е. между двумя распределениями существует точное соответствие, величина $D_{f||g}$ равна нулю.

2. Она инвариантна к следующим изменениям компонентов вектора $\mathbf{x}$.
   - Любое изменение порядка компонентов вектора.
   - Масштабирование амплитуды.
   - Монотонные нелинейные преобразования.

Взаимная информация $I(\mathbf{X};\mathbf{Y})$ между парой векторов $\mathbf{X}$ и $\mathbf{Y}$ имеет интересную интерпретацию в терминах дивергенции Кулбека–Лейблера. Прежде всего заметим, что

$$f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) = f_{\mathbf{Y}}(\mathbf{y}|\mathbf{x})f_{\mathbf{X}}(\mathbf{x}). \tag{10.37}$$

Исходя из этого, формулу (10.35) можно переписать в эквивалентном виде:

$$I(\mathbf{X};\mathbf{Y}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) \log\left(\frac{f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})}{f_{\mathbf{X}}(\mathbf{x})f_{\mathbf{Y}}(\mathbf{y})}\right) d\mathbf{x}d\mathbf{y}.$$

Сравнивая эту формулу с (10.36), можно сразу же прийти к следующему результату:

$$I(\mathbf{X};\mathbf{Y}) = D_{f_{\mathbf{X},\mathbf{Y}}||f_{\mathbf{X}}f_{\mathbf{Y}}}. \tag{10.38}$$

Образно говоря, взаимная информация $I(\mathbf{X};\mathbf{Y})$ между $\mathbf{X}$ и $\mathbf{Y}$ равна дивергенции Кулбека–Лейблера между функцией плотности совместной вероятности $f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})$ и произведением функций плотности вероятности $f_X(\mathbf{x})$ и $f_Y(\mathbf{y})$.

Частным случаем последнего результата является дивергенция Кулбека–Лейблера между функцией плотности вероятности $f_{\mathbf{X}}(\mathbf{x})$ случайного вектора $\mathbf{X}$ размерности $m \times 1$ и произведением m его граничных функций плотности вероятности. Пусть $\tilde{f}_{X_i}(x_i)$ – i-я граничная функция плотности вероятности элемента X_i:

$$\tilde{f}_{X_i}(x_i) = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x})d\mathbf{x}^{(i)}, \quad i = 1, 2, \ldots, m, \tag{10.39}$$

где $\mathbf{x}^{(i)}$ – вектор размерности $(m-1) \times 1$, оставшийся после удаления из вектора $\mathbf{x}$ его i-го элемента. Дивергенция Кулбека–Лейблера между $f_{\mathbf{X}}(\mathbf{x})$ и *факториальным распределением* (factorial distribution) $\prod_i \tilde{f}_{X_i}(x_i)$ будет иметь следующий вид:

$$D_{f_{\mathbf{X}}||\tilde{f}_{\mathbf{X}}} = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{f_{\mathbf{X}}(\mathbf{x})}{\prod_{i=1}^{m} \tilde{f}_{X_i}(x_i)}\right) d\mathbf{x}, \tag{10.40}$$

Это выражение можно переписать в расширенной форме:

$$D_{f_{\mathbf{X}}||\tilde{f}_{\mathbf{X}}} = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} - \sum_{i=1}^{m} \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log \tilde{f}_{x_i}(x_i) d\mathbf{x}. \tag{10.41}$$

Первый интеграл в правой части равенства (10.41) равен по определению $-h(\mathbf{X})$, где $h(\mathbf{X})$ — дифференциальная энтропия $\mathbf{X}$. Для того чтобы преобразовать второе слагаемое, заметим, что

$$d\mathbf{x} = d\mathbf{x}^{(i)} dx_i.$$

Исходя из этого, можно записать:

$$\int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log \tilde{f}_{X_i}(x_i) d\mathbf{x} = \int_{-\infty}^{\infty} \log \tilde{f}_{X_i}(x_i) \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}^{(i)} dx_i, \tag{10.42}$$

где внутренний интеграл в правой части берется по вектору $\mathbf{x}^{(i)}$ размерности $(m-1)\times 1$, а внешний — по скаляру x_i. Однако из выражения (10.39) видно, что внутренний интеграл равен граничной функции плотности вероятности $\tilde{f}_{X_i}(x_i)$. Следовательно, выражение (10.42) можно переписать в эквивалентном виде:

$$\begin{aligned} \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log \tilde{f}_{X_i}(x_i) d\mathbf{x} &= \int_{-\infty}^{\infty} \tilde{f}_{X_i}(x_i) \log \tilde{f}_{X_i}(x_i) dx_i = \\ &= -\tilde{h}(X_i), \quad i = 1, 2, \ldots, m, \end{aligned} \tag{10.43}$$

где $-\tilde{h}(X_i)$ — i-я *граничная энтропия* (marginal entropy) (т.е. дифференциальная энтропия, основанная на граничной функции плотности вероятности $\tilde{f}_{X_i}(x_i)$). В заключение, подставив (10.43) в (10.41) и учитывая, что первый интеграл в (10.41) равен $-h(\mathbf{X})$, формулу дивергенции Кулбека–Лейблера можно упростить:

$$D_{f_{\mathbf{X}}||\tilde{f}_{\mathbf{X}}} = -h(\mathbf{X}) + \sum_{i=1}^{m} \tilde{h}(X_i). \tag{10.44}$$

Эту формулу далее в этой главе мы будем использовать при изучении задачи слепого разделения источников.

Декомпозиция Пифагора

Теперь рассмотрим дивергенцию Кулбека–Лейблера между функциями плотности вероятности $f_{\mathbf{X}}(\mathbf{x})$ и $f_{\mathbf{U}}(\mathbf{x})$. Случайный вектор $\mathbf{U}$ размерности $m \times 1$ состоит из неза-

висимых переменных:

$$f_{\mathbf{U}}(\mathbf{x}) = \prod_{i=1}^{m} f_{U_i}(x_i),$$

а случайный вектор $\mathbf{X}$ размерности $m \times 1$ определяется в терминах $\mathbf{U}$ как

$$\mathbf{X} = \mathbf{AU},$$

где $\mathbf{A}$ — некоторая недиагональная матрица. Пусть $\tilde{f}_{X_i}(x_i)$ — граничная функция плотности вероятности каждого компонента X_i для $f_X(\mathbf{x})$. Тогда дивергенция Кулбека–Лейблера между $f_X(\mathbf{x})$ и $f_U(\mathbf{x})$ допускает следующую декомпозицию Пифагора:

$$D_{f_{\mathbf{X}}||f_{\mathbf{U}}} = D_{f_{\mathbf{X}}||\tilde{f}_{\mathbf{X}}} + D_{\tilde{f}_{\mathbf{X}}||f_{\mathbf{U}}}. \tag{10.45}$$

Мы называем это классическое соотношение декомпозицией Пифагора, так как оно имеет информационно-геометрическую интерпретацию[8] [27].

---

[8] Для доказательства декомпозиции (10.45) можно поступить следующим образом. По определению

$$D_{f_{\mathbf{X}}||f_{\mathbf{U}}} = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{f_{\mathbf{X}}(\mathbf{x})}{f_{\mathbf{U}}(\mathbf{x})}\right) d\mathbf{x} = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{f_{\mathbf{X}}(\mathbf{x})}{\tilde{f}_{\mathbf{X}}(\mathbf{x})}\frac{\tilde{f}_{\mathbf{X}}(\mathbf{x})}{f_{\mathbf{U}}(\mathbf{x})}\right) d\mathbf{x} =$$
$$= \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{f_{\mathbf{X}}(\mathbf{x})}{\tilde{f}_{\mathbf{X}}(\mathbf{x})}\right) d\mathbf{x} + \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{\tilde{f}_{\mathbf{X}}(\mathbf{x})}{f_{\mathbf{U}}(\mathbf{x})}\right) d\mathbf{x} = \tag{1}$$
$$= D_{f_{\mathbf{X}}||\tilde{f}_{\mathbf{X}}} + \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{\tilde{f}_{\mathbf{X}}(\mathbf{x})}{f_{\mathbf{U}}(\mathbf{x})}\right) d\mathbf{x}.$$

Из определения $\tilde{f}_{\mathbf{X}}(\mathbf{x}) f_U(\mathbf{u})$ известно, что

$$\log\left(\frac{\tilde{f}_{\mathbf{X}}(\mathbf{x})}{f_{\mathbf{U}}(\mathbf{x})}\right) = \log\left(\frac{\prod_{i=1}^{m} \tilde{f}_{X_i}(x_i)}{\prod_{i=1}^{m} f_{U_i}(x_i)}\right) = \sum_{i=1}^{m} \log\left(\frac{\tilde{f}_{X_i}(x_i)}{f_{U_i}(x_i)}\right).$$

Обозначим символом B интеграл в последней строке формулы (1). Тогда можно записать:

$$B = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{\tilde{f}_{\mathbf{X}}(\mathbf{x})}{f_{\mathbf{U}}(\mathbf{x})}\right) d\mathbf{x} = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{\prod_{i=1}^{m} \tilde{f}_{X_i}(x_i)}{\prod_{i=1}^{m} f_{U_i}(x_i)}\right) d\mathbf{x} =$$
$$= \sum_{i=1}^{m} \int_{-\infty}^{\infty} \left(\log\left(\frac{\tilde{f}_{X_i}(x_i)}{f_{U_i}(x_i)}\right) \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}^{(i)}\right) dx_i = \tag{2}$$
$$= \sum_{i=1}^{m} \int_{-\infty}^{\infty} \log\left(\frac{\tilde{f}_{X_i}(x_i)}{f_{U_i}(x_i)}\right) \tilde{f}_{X_i}(x_i) dx_i,$$

где в последней строке использовалось определение (10.39). Интеграл (2) в своем окончательном виде является дивергенцией Кулбека–Лейблера $D_{\tilde{f}_{X_i}||f_{U_i}}$ $i = 1, 2, \ldots, m$. Для того чтобы привести выражение для B к его окончательному виду, заметим, что площадь области под графиком $\tilde{f}_{X_i}(x_i)$ равна единице. Исходя из этого, можно записать:

$$B = \sum_{i=1}^{m} \int_{-\infty}^{\infty} \prod_{j=1}^{m} \tilde{f}_{X_j}(x_j) \left(\log\left(\frac{\tilde{f}_{X_i}(x_i)}{f_{U_i}(x_i)}\right) dx_i\right) \tilde{f}_{X_i}(x_i) d\mathbf{x}^{(i)} =$$
$$= \int_{-\infty}^{\infty} \tilde{f}_{\mathbf{X}}(\mathbf{x}) \log\left(\frac{\prod_{i=1}^{m} \tilde{f}_{X_i}(x_i)}{\prod_{i=1}^{m} f_{U_i}(x_i)}\right) d\mathbf{x} = \tag{3}$$
$$= D_{\tilde{f}_{\mathbf{X}}||f_{\mathbf{U}}},$$

где в первой строке использовалось определение $d\mathbf{x}=dx_i\ d\mathbf{x}^{(i)}$ (см. раздел 10.5). Подставляя (3) в (1), получим искомую декомпозицию:

$$D_{f_{\mathbf{X}}||f_{\mathbf{U}}} = D_{f_{\mathbf{X}}||\tilde{f}_{\mathbf{X}}} + D_{\tilde{f}_{\mathbf{X}}||f_{\mathbf{U}}}.$$

10.6. Взаимная информация как оптимизируемая целевая функция

После рассмотрения основ теории информации Шеннона можно начать обсуждение ее роли в изучении самоорганизующихся систем.

Чтобы открыть эту дискуссию, рассмотрим нейронную сеть с множеством входов и выходов. Главная цель — добиться самоорганизации этой системы для выполнения поставленной задачи (например, моделирования, извлечения статистически значимых признаков, разделения сигнала). Этого можно добиться, выбрав взаимную информацию между определенными переменными системы в качестве оптимизируемой *целевой функции* (objective function). Этот конкретный выбор был обусловлен следующими соглашениями.

- Взаимная информация имеет ряд уникальных свойств, о которых уже говорилось в разделе 10.4.
- Она может быть определена без необходимости использования учителя, так что основное условие самоорганизации соблюдено.

Таким образом, задача состоит в такой настройке свободных параметров (т.е. синаптических весов) системы, чтобы оптимизировать взаимную информацию.

В зависимости от области применения можно идентифицировать четыре различных сценария (рис. 10.2), которые могут возникнуть на практике. Эти сценарии можно описать следующим образом.

- В сценарии 1 (см. рис. 10.2, *а*) входной вектор $\mathbf{X}$ состоит из элементов $X_1, X_2, \ldots, X_m$, а выходной вектор $\mathbf{Y}$ — из элементов $Y_1, Y_2, \ldots, Y_l$. *Требуется максимизировать информацию о входе системы* $\mathbf{X}$, *передаваемую на выход системы* $\mathbf{Y}$.
- В сценарии 2 (см. рис. 10.2, *б*) пара входных векторов $\mathbf{X}_a$ и $\mathbf{X}_b$ порождена смежными, но не пересекающихся областями образа. Входы $\mathbf{X}_a$ и $\mathbf{X}_b$ производят скалярные выходы Y_a и Y_b соответственно. *Требуется максимизировать информацию об* Y_a, *передаваемую на выход* Y_b, *и наоборот.*
- В сценарии 3 (см. рис. 10.2, *в*) входные векторы $\mathbf{X}_a$ и $\mathbf{X}_b$ порождены соответствующей парой областей, принадлежащих разным образам. Эти входные векторы производят скалярные выходы Y_a и Y_b соответственно. *Требуется минимизировать информацию об* Y_b, *передаваемую на выход* Y_a.
- В сценарии 4 (см. рис. 10.2, *г*) входной вектор $\mathbf{X}$ и выходной вектор $\mathbf{Y}$ определяются аналогично рис. 10.2, *а*, но в данном случае имеют место равные размерности (т.е. $l = m$). *Требуется минимизировать статистическую зависимость между компонентами выходного вектора* $\mathbf{Y}$.

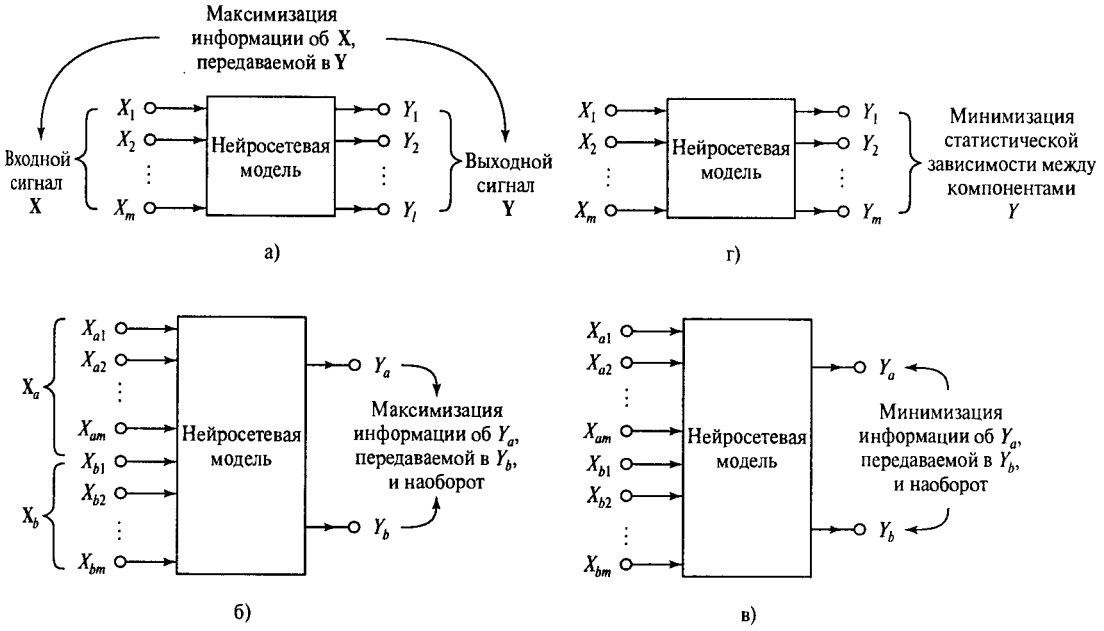


Рис. 10.2. Четыре основных сценария применения принципа максимума взаимной информации (Infomax) и его трех вариантов

Во всех этих ситуациях главную роль играет взаимная информация. Однако способ ее формулировки во многом зависит от особенностей конкретной ситуации. В оставшейся части настоящей главы мы рассмотрим описанные выше сценарии и их практическое применение. При этом последовательность изложения будет соответствовать только что представленному порядку.

10.7. Принцип максимума взаимной информации

Идея создания нейронного процессора, максимизирующего взаимную информацию $I(\mathbf{X};\mathbf{Y})$, уходит корнями в основы статистической обработки сигнала. Этот метод оптимизации включен в *принцип максимума взаимной информации* (maximum mutual information или Infomax) Линскера [651], [653], [655], который можно сформулировать следующим образом.

Преобразование случайного вектора **X**, *наблюдаемого на входном слое нейронной системы, в случайный вектор* **Y**, *наблюдаемый на выходе той же системы, должно выбираться таким образом, чтобы совместная работа нейронов выходного слоя максимизировала информацию о деятельности входного слоя. Максимизируемой целевой функцией при этом является взаимная информация* $I(\mathbf{X};\mathbf{Y})$ *между векторами* **X** *и* **Y**.

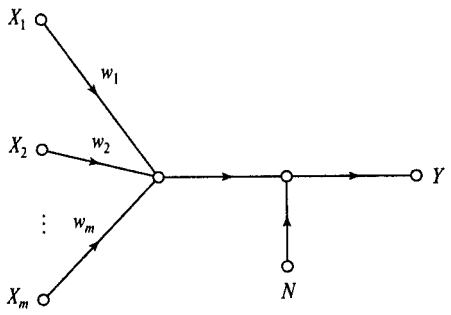


Рис. 10.3. Граф передачи сигнала в зашумленном нейроне

Принцип Infomax предоставляет математическую среду для самоорганизации систем передачи сигнала, показанных на рис. 10.2, *а*, которая не зависит от правил реализации. Этот принцип можно также рассматривать как нейросетевую составляющую концепции *емкости канала* (channel capacity), которая определяет предел Шеннона объема информации, передаваемой по каналу связи.

Далее мы проиллюстрируем применение принципа Infomax двумя примерами, в которых будет участвовать один зашумленный нейрон. В первом из этих примеров шум будет добавляться к выходному сигналу, а во втором — ко входному.

Пример 10.4

Отдельный нейрон, находящийся под действием шума

Рассмотрим простой пример линейного нейрона, получающего входной сигнал от множества, состоящего из m узлов-источников. Пусть выход этого нейрона с учетом шума выражается соотношением

$$Y = \left(\sum_{i=1}^{m} w_i X_i \right) + N, \tag{10.46}$$

где w_i — i-й синаптический вес; N — шум обработки сигнала (рис. 10.3). При этом предполагается следующее.

- Выход нейрона Y представляет собой гауссову случайную переменную с дисперсией σ_Y^2.
- Шум обработки сигнала N также является гауссовой случайной переменной с нулевым средним и дисперсией σ_N^2.
- Шум обработки является некоррелированным по всем своим входным составляющим, т.е.

$$E[NX_i] = 0 \text{ для всех } i.$$

Гауссово распределение выходного сигнала Y можно обеспечить двумя способами. Входные сигналы $X_1, X_2, \ldots, X_m$ имеют гауссово распределение. При этом предполагается, что аддитивный шум N также является гауссовым. Тогда гауссово распределение выходного сигнала Y вытекает из того факта, что он является взвешенной суммой гауссовых случайных переменных. Как альтернатива, входные сигналы $X_1, X_2, \ldots, X_m$ могут быть независимо и равномерно распределены. В этом случае распределение их взвешенной суммы становится гауссовым при больших значениях m. Это — следствие *центральной предельной теоремы* (central limit theorem).

Приступая к анализу, в первую очередь обратим внимание на вторую строку равенства (10.30). В ней взанмная информация $I(Y;\mathbf{X})$ между выходом нейрона Y и входным вектором $\mathbf{X}$ определяется следующим образом:

$$I(Y;\mathbf{X}) = h(Y) - h(Y|\mathbf{X}). \tag{10.47}$$

Посмотрев на формулу (10.46), несложно заметить, что функция плотности вероятности переменной Y для входного вектора $\mathbf{X}$ равна функции плотности вероятности суммы константы и гауссовой случайной переменной. Следовательно, условная энтропия $h(Y|\mathbf{X})$ является "информацией", которую выход Y накапливает о шуме обработки сигнала N, а не о самом векторе полезного сигнала $\mathbf{X}$. Исходя из этого, примем

$$h(Y|\mathbf{X}) = h(N)$$

и перепишем выражение (10.47) в следующем виде:

$$I(Y;\mathbf{X}) = h(Y) - h(N). \tag{10.48}$$

Применяя выражение (10.22) к дифференциальной энтропии гауссовой случайной переменной к нашей задаче, получим:

$$h(Y) = \frac{1}{2}[1 + \log(2\pi\sigma_Y^2)] \tag{10.49}$$

и

$$h(N) = \frac{1}{2}[1 + \log(2\pi\sigma_N^2)]. \tag{10.50}$$

После подстановки выражений (10.49) и (10.50) в формулу (10.48) и упрощения получим:

$$I(Y;\mathbf{X}) = \frac{1}{2}\log\left(\frac{\sigma_Y^2}{\sigma_N^2}\right), \tag{10.51}$$

где σ_Y^2 зависит от σ_N^2.

Частное σ_Y^2/σ_N^2 можно рассматривать как *отношение сигнал/шум* (signal-to-noise ratio). Предполагая фиксированность дисперсии σ_N^2, можно заметить, что взаимная информация $I(Y;\mathbf{X})$ достигает максимума при максимизации дисперсии σ_Y^2 выхода нейрона Y. Таким образом, можно утверждать, что при определенных условиях максимизация дисперсии выходного сигнала максимизирует взаимную информацию между входным и выходным сигналами нейрона [653]. ■

Пример 10.5

Отдельный нейрон, входной сигнал которого искажен аддитивным шумом

Теперь предположим, что шум, искажающий поведение нейрона, поступает на его вход через синаптические связи (рис. 10.4). Согласно этой модели шума:

$$Y = \sum_{i=1}^{m} w_i(X_i + N_i), \tag{10.52}$$

где предполагается, что каждый шум N_i является независимой гауссовой случайной переменной с нулевым средним и дисперсией σ_N^2. Тогда формулу (10.52) можно переписать в виде, аналогичном (10.46):

$$Y = \left(\sum_{i=1}^{m} w_i X_i\right) + N',$$

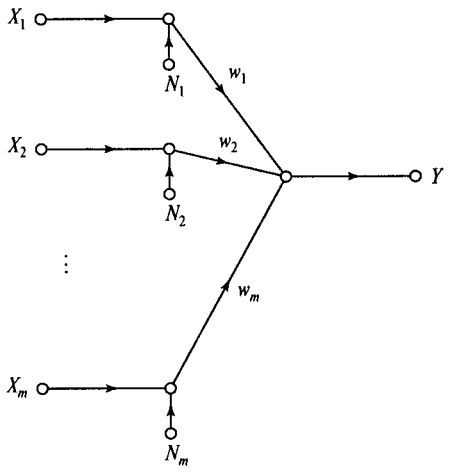


Рис. 10.4. Вторая модель шума

где N' — составной компонент шума, имеющий вид.

$$N' = \sum_{i=1}^{m} w_i N_i.$$

Составной шум N' имеет гауссово распределение с нулевым средним и дисперсией, равной сумме дисперсий отдельных компонентов шума, т.е.

$$\sigma_N'^2 = \sum_{i=1}^{m} w_i^2 \sigma_N^2.$$

Как и раньше, предположим, что выход Y нейрона имеет гауссово распределение с дисперсией σ_Y^2. Взаимная информация $I(Y;\mathbf{X})$ между Y и $\mathbf{X}$ все так же задается формулой (10.47). Однако на этот раз условная энтропия $h(Y|\mathbf{X})$ определяется по-другому:

$$h(Y|\mathbf{X}) = h(N') = \frac{1}{2}(1 + 2\pi\sigma_N'^2) = \frac{1}{2}\left[1 + 2\pi\sigma_N^2 \sum_{i=1}^{m} w_i^2\right]. \tag{10.53}$$

Таким образом, подставляя выражения (10.49) и (10.53) в (10.47), получим [653]:

$$I(Y;\mathbf{X}) = \frac{1}{2}\log\left(\frac{\sigma_Y^2}{\sigma_N^2 \sum_{i=1}^{m} w_i^2}\right). \tag{10.54}$$

Если дисперсия σ_N^2 является константой, взаимная информация $I(Y;\mathbf{X})$ достигает максимума при максимизации отношения $\sigma_Y^2 / \sum_{i=1}^{m} w_i^2$, где σ_Y^2 — функция от w_i. ■

Что полезного можно извлечь из примеров 10.4 и 10.5?

Первое, что можно увидеть в двух выше приведенных примерах, — это то, что результат применения принципа Infomax зависит от поставленной задачи. Эквивалентность между максимизацией взаимной информации $I(Y;\mathbf{X})$ и дисперсией выходного сигнала в модели на рис. 10.3 для наперед заданной дисперсии σ_N^2 не присутствует в модели на рис. 10.4. Эти две модели ведут себя одинаково только в том случае, когда на модель, показанную на рис. 10.4, наложено дополнительное условие $\sum_i w_i^2 = 1$.

В общем случае определение взаимной информации между входом **X** и выходом **Y** является достаточно сложной задачей. В примерах 10.4 и 10.5 мы провели математический анализ с предположением о том, что распределение шума в системе с одним или несколькими его источниками является *многомерным гауссовым*. Это допущение еще нужно обосновать.

При адаптации гауссовой модели шума, в сущности, вводится "суррогатная" взаимная информация в предположении, что выходной вектор нейрона **Y** имеет многомерное гауссово распределение с тем же вектором средних значений и матрицей ковариации, что и исследуемое распределение. В [647] для обоснования использования такой суррогатной взаимной информации использовалась дивергенция Кулбека–Лейблера. При этом предполагалось, что сеть хранит информацию о векторе средних значений и матрице ковариации выходного вектора **Y** и не хранит статистику более высокого порядка.

В заключение отметим, что анализ, представленный в примерах 10.4 и 10.5, проводился в контексте одного нейрона. Это было сделано преднамеренно, исходя из следующей точки зрения. Для обеспечения математической трактовки принципа Infomax оптимизация должна проводиться на уровне локального нейрона. Такая оптимизация является совместимой с сущностью самоорганизации.

Пример 10.6

В примерах 10.4 и 10.5 рассматривались зашумленные нейроны. В этом примере мы сосредоточим внимание на сети без шума, которая преобразует случайный вектор **X** с произвольным распределением в другой случайный вектор **Y** с другим распределением. Вспоминая свойство симметричности взаимной информации ($I(\mathbf{X};\mathbf{Y})=I(\mathbf{Y};\mathbf{X})$) и расширяя формулу (10.28) на описанную здесь ситуацию, взаимную информацию между входным вектором **X** и выходным вектором **Y** можно выразить следующим образом:

$$I(\mathbf{Y};\mathbf{X}) = H(\mathbf{Y}) - H(\mathbf{Y}|\mathbf{X}),$$

где $H(\mathbf{Y})$ — энтропия **Y**; $H(\mathbf{Y}|\mathbf{X})$ — условная энтропия **Y** для данного **X**. В предположении об отсутствии шума при отображении **X** на **Y** условная энтропия $H(\mathbf{Y}|\mathbf{X})$ достигает своего наименьшего значения — она расходится до $-\infty$. Этот результат вызван дифференциальной природой энтропии непрерывной случайной переменной (см. раздел 10.2). Однако эта сложность не имеет последствий, если рассматривать *градиент* взаимной информации $I(\mathbf{Y};\mathbf{X})$ по отношению к матрице весов **W**, параметризующей сеть, выполняющую отображение. В частности, можно записать, что

$$\frac{\partial I(\mathbf{Y};\mathbf{X})}{\partial \mathbf{W}} = \frac{\partial H(\mathbf{Y})}{\partial \mathbf{W}}, \tag{10.55}$$

поскольку условная энтропия $H(\mathbf{Y}|\mathbf{X})$ не зависит от **W**. Уравнение (10.55) показывает, что в сети без учета шума, осуществляющей отображение, максимизация энтропии выхода **Y** эквивалентна максимизации взаимной информации между **Y** и входом системы **X**. При этом обе максимизации производятся по отношению к матрице весов **W** сети [116]. ■

10.8. Принцип Infomax и уменьшение избыточности

В теории информации Шеннона порядок и структура представляют *избыточность*, которая уменьшает неопределенность, разрешаемую за счет получения информации. Чем больший порядок и структуру имеет исследуемый процесс, тем меньше информации мы получаем из наблюдения за этим процессом. Рассмотрим пример в наивысшей мере структурированной и избыточной последовательности символов *аааааа*. При получении первого примера *а* мы можем сразу же сказать, что оставшиеся пять примеров будут такими же. Информация, переданная такой последовательностью примеров, равна информации, содержащейся только в одном примере. Другими словами, чем более избыточной является последовательность примеров, тем меньше информации мы получаем о характеристиках среды.

Из определения взаимной информации $I(\mathbf{Y}; \mathbf{X})$ мы знаем, что она является мерой неопределенности о выходе системы **Y**, которая разрешается наблюдением за входом системы **X**. Принцип Infomax связан с максимизацией взаимной информации $I(\mathbf{Y}; \mathbf{X})$. В результате, наблюдая за входом **X**, мы получаем максимум информации о выходе системы **Y**. В свете ранее упомянутой взаимосвязи между объемом информации и избыточностью можно утверждать, что принцип Infomax ведет к уменьшению избыточности в выходе **Y** по сравнению со входом **X**.

Наличие шума в сигнале предполагает использование избыточности и других методов разнообразия [653]. Если уровень аддитивного шума во входном сигнале высок, можно использовать избыточность для борьбы с разлагающим влиянием шума. В такой среде наиболее коррелированные компоненты объединяются в процессоре для обеспечения точного представления входного сигнала. Кроме того, если уровень шума на выходе (т.е. шум процессора) достаточно высок, большая часть компонентов выходного сигнала направляется процессором для обеспечения избыточности информации. Таким образом, уменьшается количество наблюдаемых на выходе процессора независимых характеристик, однако при этом увеличивается точность представления каждой из них. Таким образом, можно утверждать, что *более высокий уровень шума приводит к необходимости избыточности в представлении*. Однако если *уровень шума низок, разнообразие представления имеет преимущество* перед избыточностью. Под разнообразием здесь понимаются два или более выходов процессора с отличными свойствами. В задаче 10.6 с точки зрения принципа Infomax обсуждается баланс между разнообразием и избыточностью. Хочется заметить, что такой баланс аналогичен балансу между смещением и дисперсией, о котором говорилось в главе 2.

Моделирование систем восприятия

С самых первых дней развития теории информации предполагалось, что избыточность сенсорных сигналов (возбуждений) важна для понимания восприятия [83], [92].

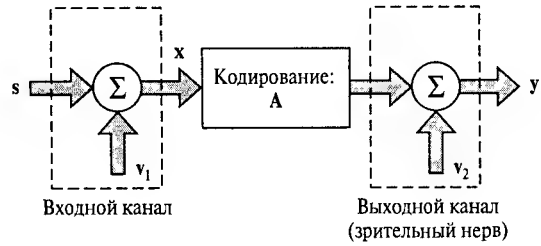


Рис. 10.5. Модель системы восприятия. Векторы сигнала **s** и векторы шумов $\mathbf{v}_1$ и $\mathbf{v}_2$ являются соответственно реализациями случайных векторов **S**, $\mathbf{N}_1$ и $\mathbf{N}_2$

И в самом деле, избыточность сенсорных "сообщений" обеспечивает знания, позволяющие мозгу строить свои "карты познания" или "рабочие модели" окружающего мира [90]. Закономерности в сенсорных сообщениях должны некоторым образом кодироваться в мозге, чтобы он понимал, что происходит. Однако *уменьшение избыточности* является более специфичной формой *гипотезы Барлоу* (Barlow's hypothesis). Эта гипотеза утверждает, что целью ранней обработки является преобразование в высшей мере избыточного сенсорного входа в более эффективный *факториальный код* (factorial code). Другими словами, нейронные выходы становятся *статистически независимыми*, когда обусловлены входом.

Под воздействием гипотезы Барлоу в [79] был сформулировал *принцип минимума избыточности* (principle of minimum redundancy) как базис информационно-теоретический модели системы восприятия (рис. 10.5). Эта модель состоит из трех компонентов: *входного канала* (input channel), *системы кодирования* (recoding system) и *выходного канала* (output channel). Выходной и входной каналы описываются следующим образом:

$$\mathbf{X} = \mathbf{S} + \mathbf{N}_1,$$

где **S** — идеальный сигнал, получаемый входным каналом; $\mathbf{N}_1$ — совокупный шум на входе. Сигнал **X**, в свою очередь, преобразовывается (кодируется) линейным матричным оператором **A**, а результат передается по оптическому нерву, или выходному каналу, и формирует выход системы:

$$\mathbf{Y} = \mathbf{AX} + \mathbf{N}_2,$$

где $\mathbf{N}_2$ — внутренний шум, налагаемый после кодирования. В подходе, предпринятом в [79], отмечалось, что световой сигнал в сетчатке содержит полезную сенсорную информацию в крайне избыточной форме. Более того, выдвигалась гипотеза, что целью обработки сигнала сетчаткой является уменьшение (или полная ликвидация) излишних битов данных как в корреляции, так и в шуме перед посылкой этого сигнала по оптическому нерву.

Для определения *меры избыточности* была введена величина

$$R = 1 - \frac{I(\mathbf{Y};\mathbf{S})}{C(\mathbf{Y})}, \tag{10.56}$$

где $I(\mathbf{Y};\mathbf{S})$ — взаимная информация между $\mathbf{Y}$ и $\mathbf{S}$; $C(\mathbf{Y})$ — мощность (выходного) канала оптического нерва. Равенство (10.56) можно объяснить с тех позиций, что мозг интересует исключительно идеальный сигнал $\mathbf{S}$, в то время как сигнал передается физически по оптическому нерву. Здесь предполагается, что при отображении входа на выход не происходит уменьшения размерности, т.е. $C(\mathbf{Y}) > I(\mathbf{Y};\mathbf{S})$. Это требование заключается в поиске такого отображения входа на выход (т.е. матрицы $\mathbf{A}$), которое минимизирует меру избыточности R при отсутствии потери информации:

$$I(\mathbf{Y};\mathbf{X}) = I(\mathbf{X};\mathbf{X}) - \varepsilon,$$

где ε — некоторый малый положительный параметр. *Мощность канала* (channel capacity) $C(\mathbf{Y})$ определяется как максимальный поток информации, который может быть передан по оптическому нерву. При этом предполагается возможность использования любых распределений входного сигнала и сохранение средней мощности входа на фиксированном уровне.

Если вектор сигнала $\mathbf{S}$ и вектор выхода $\mathbf{Y}$ имеют одну и ту же размерность, а шум в системе отсутствует, принцип минимума избыточности и принцип Infomax являются математически эквивалентными (предполагается, что в обоих случаях на вычислительную мощность выходных нейронов налагаются сходные ограничения). Для примера предположим, что мощность канала измеряется в терминах динамического диапазона выходного сигнала каждого из нейронов на рис. 10.5. Тогда, согласно принципу минимума избыточности, минимизируемой величиной является

$$1 - \frac{I(\mathbf{Y};\mathbf{S})}{C(\mathbf{Y})}$$

для данной допустимой потери информации и, таким образом, для данного $I(\mathbf{Y};\mathbf{S})$. Таким образом, минимизируемую величину можно записать следующим образом:

$$F_1(\mathbf{Y};\mathbf{S}) = C(\mathbf{Y}) - \lambda I(\mathbf{Y};\mathbf{S}). \tag{10.57}$$

С другой стороны, согласно принципу Infomax, максимизируемая величина в модели на рис. 10.5 выглядит следующим образом:

$$F_2(\mathbf{Y};\mathbf{S}) = I(\mathbf{Y};\mathbf{S}) + \lambda C(\mathbf{Y}). \tag{10.58}$$

Несмотря на то что функции $F_1(\mathbf{Y};\mathbf{S})$ и $F_2(\mathbf{Y};\mathbf{S})$ отличаются друг от друга, их оптимизация приводит к одному и тому же результату — они формулируют методы множителей Лагранжа, в которых роли $I(\mathbf{Y};\mathbf{S})$ и $C(\mathbf{Y})$ просто меняются местами.

В этом обсуждении важно заметить, что, несмотря на различия в формулировке, эти два информационно-теоретических принципа приводят к сходным результатам. В итоге можно сказать следующее: максимизация взаимной информации между входом и выходом нейронной системы и в самом деле приводит к уменьшению избыточности[9].

10.9. Пространственно связные признаки

Принцип Infomax, сформулированный в разделе 10.6, применяется в ситуациях, когда максимизируемой целевой функцией является взаимная информация $I(\mathbf{Y};\mathbf{X})$ между выходным вектором $\mathbf{Y}$ и входным вектором $\mathbf{X}$ (см. рис. 10.2, *а*). После внесения соответствующих изменений в терминологию этот принцип можно расширить для обработки изображений без учителя [114]. Необработанный пиксель такого изображения содержит массу информации об обрабатываемом изображении. В частности, на интенсивность каждого пикселя влияют такие внутренние параметры изображения, как глубина, отражающая способность и ориентация поверхности, равно как фоновый шум и освещенность. Нашей целью является создание такой системы самоорганизации, которая способна через обучение преобразовывать эту сложную информацию в более простую форму. Конкретизируя цель, ее можно описать как извлечение признаков более высокого порядка, которые представляют *простую связность пространства* (simple coherence across space) таким способом, чтобы представление информации в пространственно-локализованной области изображения облегчало ее представление в соседних областях. При этом под областью понимается некоторый набор пикселей изображения. Описанный здесь сценарий продемонстрирован на рис. 10.2, *б*.

Исходя из этого, можно сформулировать *первый вариант* принципа Infomax[10] [110], [114].

---

[9] В [766], [767] также рассматривалась связь между принципом Infomax и уменьшением избыточности. Авторы пришли к аналогичному выводу о том, что максимизация взаимной информации между входным и выходным векторами нейронной системы ведет к уменьшению объема данных. В [407] рассматривается применение фильтров Infomax к сетчатке глаза. В ней показано, что избыточность существенна для достижения робастности к шуму во внутреннем представлении среды, которая наблюдалась в сенсорных системах, подобных сетчатке.

[10] В [114] для обозначения первого варианта принципа Infomax использовалось обозначение I_{max}.

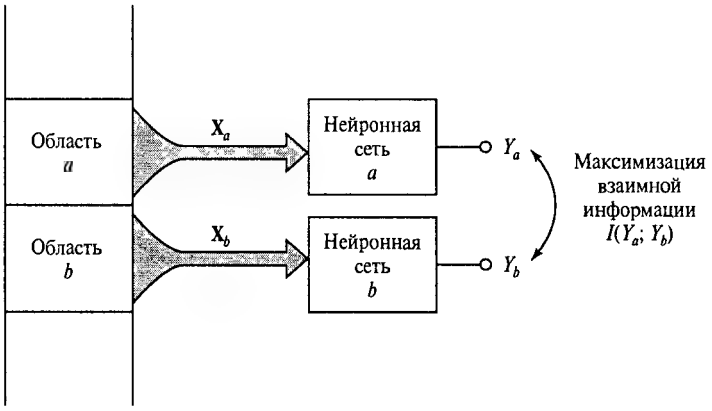


Рис. 10.6. Обработка двух смежных областей изображения в соответствии с первым вариантом принципа Infomax

Преобразование пары векторов $\mathbf{X}_a$ и $\mathbf{X}_b$ (представляющих смежные, непересекающиеся области изображения в нейронной системе) должно выбираться таким образом, чтобы скалярный выход Y_a, произведенный в ответ на входное воздействие $\mathbf{X}_a$, максимизировал информацию о втором скалярном выходе Y_b, произведенном в ответ на входной сигнал $\mathbf{X}_b$. Максимизируемой функцией при этом является взаимная информация $I(Y_a; Y_b)$ между выходами Y_a и Y_b

Этот принцип рассматривается как один из вариантов принципа Infomax не потому, что он эквивалентен последнему или выводится из него, а потому, что они пропитаны общим духом.

Для примера рассмотрим рис. 10.6, на котором показаны две нейронные сети (модуля), a и b, которые получают входные сигналы $\mathbf{X}_a$ и $\mathbf{X}_b$ из смежных, неперекрывающихся областей некоторого изображения. Скалярами Y_a и Y_b обозначим выходы этих двух модулей, соответствующие входным векторам $\mathbf{X}_a$ и $\mathbf{X}_b$. Обозначим символом S компонент сигнала, который является общим для Y_a и Y_b. Он выражает собой пространственную связность двух соответствующих областей исходного изображения. Выходы Y_a и Y_b можно выразить как зашумленные версии общего сигнала S:

$$Y_a = S + N_a \tag{10.59}$$

и

$$Y_b = S + N_b, \tag{10.60}$$

где N_a и N_b — компоненты аддитивного шума, которые предполагаются статистически независимыми гауссовыми случайными переменными с нулевым средним. Для компонента S также предполагается гауссово распределение. В соответствии с (10.59) и (10.60) два модуля — a и b (см. рис. 10.6) — имеют соответствующие допущения относительно друг друга.

Используя последнюю строку выражения (10.30), взаимную информацию между Y_a и Y_b можно выразить следующим образом:

$$I(Y_a; Y_b) = h(Y_a) + h(Y_b) - h(Y_a, Y_b). \tag{10.61}$$

Согласно формуле (10.22) для дифференциальной энтропии гауссовых случайных переменных, величину $h(Y_a)$ можно выразить соотношением

$$h(Y_a) = \frac{1}{2}\left[1 + \log\left(2\pi\sigma_a^2\right)\right], \tag{10.62}$$

где σ_a^2 — дисперсия переменной Y_a. Аналогично, дифференциальная энтропия $h(Y_b)$ имеет следующий вид:

$$h(Y_b) = \frac{1}{2}\left[1 + \log\left(2\pi\sigma_b^2\right)\right], \tag{10.63}$$

где σ_b^2 — дисперсия переменной Y_b. Используя формулу (10.24), мы можем описать совместную дифференциальную энтропию $h(Y_a, Y_b)$:

$$h(Y_a, Y_b) = 1 + \log(2\pi) + \frac{1}{2}\log\left|\det(\boldsymbol{\Sigma})\right|. \tag{10.64}$$

Матрица $\boldsymbol{\Sigma}$ размерности 2×2 является матрицей ковариации Y_a и Y_b, определяемой следующим образом:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_a^2 & \rho_{ab}\sigma_a\sigma_b \\ \rho_{ab}\sigma_a\sigma_b & \sigma_b^2 \end{bmatrix}, \tag{10.65}$$

где ρ_{ab} — *коэффициент корреляции* Y_a и Y_b:

$$\rho_{ab} = \frac{E[(Y_a - E[Y_a])(Y_b - E[Y_b])]}{\sigma_a\sigma_b}. \tag{10.66}$$

Исходя из этого, определитель матрицы $\boldsymbol{\Sigma}$ имеет следующее значение:

$$\det(\boldsymbol{\Sigma}) = \sigma_a^2\sigma_b^2(1 - \rho_{ab}^2), \tag{10.67}$$

и выражение (10.64) можно переписать следующим образом:

$$h(Y_a, Y_b) = 1 + \log(2\pi) + \frac{1}{2}\log\left[\sigma_a^2\sigma_b^2(1 - \rho_{ab}^2)\right]. \tag{10.68}$$

Подставляя выражения (10.62), (10.63) и (10.68) в (10.61) и упрощая результат, получим:

$$I(Y_a; Y_b) = -\frac{1}{2}\log(1 - \rho_{ab}^2). \tag{10.69}$$

Из выражения (10.69) ясно видно, что максимизация взаимной информации $I(Y_a;Y_b)$ эквивалентна максимизации коэффициента корреляции ρ_{ab}. И это интуитивно понятно. Обратим внимание, что по определению $|\rho_{ab}| \leq 1$.

Максимизация взаимной информации $I(Y_a;Y_b)$ может рассматриваться как нелинейное обобщение канонической корреляции в статистике [114]. При наличии двух входных векторов $\mathbf{X}_a$ и $\mathbf{X}_b$ (не обязательно одинаковой размерности) и двух соответствующих векторов весов $\mathbf{w}_a$ и $\mathbf{w}_b$ целью *анализа канонической корреляции* (canonical correlation analysis) является поиск линейной комбинации $Y_a = \mathbf{w}_a^T\mathbf{X}_a$ и $Y_b = \mathbf{w}_b^T\mathbf{X}_b$, имеющей максимальную корреляцию [57]. Максимизация $I(Y_a;Y_b)$ является нелинейным обобщением канонической корреляции, благодаря нелинейности, встроенной в конструкцию нейронных модулей (см. рис. 10.6).

В [114] было продемонстрировано, что с помощью максимизации взаимной информации $I(Y_a;Y_b)$ можно извлечь различия из случайных точечных стереограмм. Это сложная задача извлечения признаков, которая не может быть решена ни однослойной, ни линейной нейронной сетью.

10.10. Пространственно несвязные признаки

Обработка изображения без учителя, рассмотренная в предыдущем разделе, связана с извлечением пространственно связных признаков изображения. В этом разделе рассмотрим совершенно противоположную ситуацию. В качестве примера рассмотрим модель, представленную на рис. 10.2, *в*. В ней целью является увеличение *пространственных различий* (spatial differences) между парой соответствующих областей, взятых из разных изображений. В то время как взаимная информация между выходами модулей в модели на рис. 10.2, *б* максимизируется, здесь нужен прямо противоположный результат (см. рис. 10.2, *в*).

Таким образом, можно сформулировать второй вариант принципа Infomax [1067], [1068].

Преобразование пары векторов $\mathbf{X}_a$ и $\mathbf{X}_b$ (представляющих соответствующие области разных изображений в нейронной системе) должно выбираться таким образом, чтобы скалярный выход Y_a, произведенный в ответ на входное воздействие $\mathbf{X}_a$, минимизировал информацию о втором скалярном выходе Y_b, произведенном в ответ на входной сигнал $\mathbf{X}_b$. Минимизируемой функцией при этом является взаимная информация $I(Y_a; Y_b)$ между выходами Y_a и Y_b.

Этот принцип назван одним из вариантов принципа Infomax не потому, что они эквивалентны или проистекают один из другого, — их объединяет общий дух[11].

Второй вариант принципа Infomax нашел свое применение в *радарной поляриметрии* (radar polarimetry), где радарная система формирует пару изображений интересующего объекта, передавая поляризованный сигнал и получая отклик от объекта с той же или отличной поляризацией. Поляризация может быть вертикальной и горизонтальной. Например, может существовать пара радарных изображений, одно из которых отображает поляризацию (вертикальную или горизонтальную), а второе — перекрестную поляризацию (горизонтальную при передаче и вертикальную при приеме). Такое применение было описано в работах, посвященных задаче *расширения цели поляризации* (enhancement of polarization target) в дуально-поляризованной радарной системе [1067], [1068]. Работу радара можно описать следующим образом. Некогерентный радар посылает горизонтально поляризованный сигнал, а получает отклик по вертикально и горизонтально поляризованным каналам. Интересующий объект может *скручивать поляризацию* (polarization-twisting), т.е. является *рефлектором*, созданным для поворота поляризации на 90°. При нормальной работе радарной системы определение такой цели становится сложным за счет несовершенства системы, равно как и за счет отражения от нежелательных поляриметрических объектов на поверхности. Мы предполагаем нелинейность отображения, чтобы учесть негауссово распределение, присущее радарным откликам. Задача расширения объекта лежит в классе вариационных задач, включающих в себя минимизацию квадратичного функционала стоимости при наличии ограничений. Конечный результат является обработанным изображением с перекрестной поляризацией, который повышает видимость объекта в гораздо большей мере, чем какая-либо линейная методика типа анализа главных компонентов. Модель, использованная в рассматриваемой работе, предполагает наличие гауссовой статистики в преобразованных данных, поскольку независимая от модели оценка функции плотности вероятности является вычислительно сложной задачей. Взаимная информация между двумя гауссовыми переменными Y_a и Y_b определяется по формуле (10.61). Для того чтобы обучить синаптические веса двух моделей, используется вариационный подход. При этом требуется уменьшить зашумленность радарного сигнала, обычно возникающую в горизонтально и вертикально поляризованных сигналах радара. Чтобы удовлетворить этому требованию, взаимная информация $I(Y_a, Y_b)$ минимизируется при дополнительном условии, налагаемом на синаптические веса:

$$P = (\mathrm{tr}[\mathbf{W}^T\mathbf{W}] - 1)^2, \tag{10.70}$$

[11] В [1070] рассматривалась *отрицательная информационная магистраль* (negative information pathway) и оптимизировалась взаимная информация между сигналами на выходе и входе магистрали, взятая с противоположным знаком. Было показано, что при адаптации такая система стремится стать дискриминатором более часто встречающихся в множестве входных сигналов примеров. Эта модель слабо связана со вторым вариантом принципа Infomax.

где $\mathbf{W}$ — сводная матрица весов сети; $\mathrm{tr}[\cdot]$ — след матрицы, заключенной в квадратные скобки. Стационарная точка достигается, когда

$$\nabla_{\mathbf{W}} I(Y_a; Y_b) + \lambda \nabla_{\mathbf{W}} P = 0, \tag{10.71}$$

где λ — множитель Лагранжа. Для поиска минимума используется квазиньютоновский метод (см. главу 4).

На рис. 10.7 показана архитектура нейронной сети, использованная в [1067], [1068]. Для каждого из двух модулей была выбрана сеть гауссовых радиальных базисных функций, так как эти сети имеют одно преимущество — они реализуют множество фиксированных базисных функций (т.е. неадаптивный скрытый слой). Входные данные были расширены на базисные функции, после чего скомбинированы с использованием слоев *линейных* весов. Пунктирными линиями на рис. 10.7 показаны перекрестные связи между двумя модулями. Центры гауссовых функций выбирались из интервалов, равномерно распределенных по всему входному пространству. При этом их ширина определялась эвристически. На рис. 10.8, *а* показан ряд горизонтально и вертикально поляризованных (получаемых радаром) изображений парковой зоны озера Онтарио. Координаты по горизонтальной оси увеличиваются в направлении слева направо, а по вертикальной оси — сверху вниз. На рис. 10.8, *б* показано комбинированное изображение, полученное минимизацией взаимной информации между вертикально и горизонтально поляризованными изображениями. Яркое пятно, отчетливо видное на изображении, соответствует смешанному рефлектору, скручивающему поляризацию и расположенному вдоль берега озера. По эффективности подавления искажений описанная здесь информационно-теоретическая модель превосходит все стандартные модели, основанные на анализе главных компонентов [1067], [1068][12].

10.11. Анализ независимых компонентов

Теперь рассмотрим последний сценарий, изображенный на рис. 10.2, *г*. Для того чтобы подчеркнуть специфику представленной здесь задачи, рассмотрим блочную диаграмму, показанную на рис. 10.9. Работа начинается со случайного входного вектора $\mathbf{U}(n)$, определенного следующим образом:

$$\mathbf{U} = [U_1, U_2, \ldots, U_m]^T,$$

где m компонентов вектора выбираются из множества *независимых источников*. В данной задаче рассматриваются временные последовательности, так что аргумент

[12] Система, описанная в [1067], содержит процессор постопределения (postdetection), который использует априорную информацию о расположении рефлектора на границе между водой и сушей на берегу. *Нечеткий процессор* (fuzzy processor) комбинирует первичное обнаружение с выходом детектора, базирующегося на визуальном наблюдении. Это приводит к эффективному удалению ложных сигналов, результатом чего становится повышение производительности такой системы.

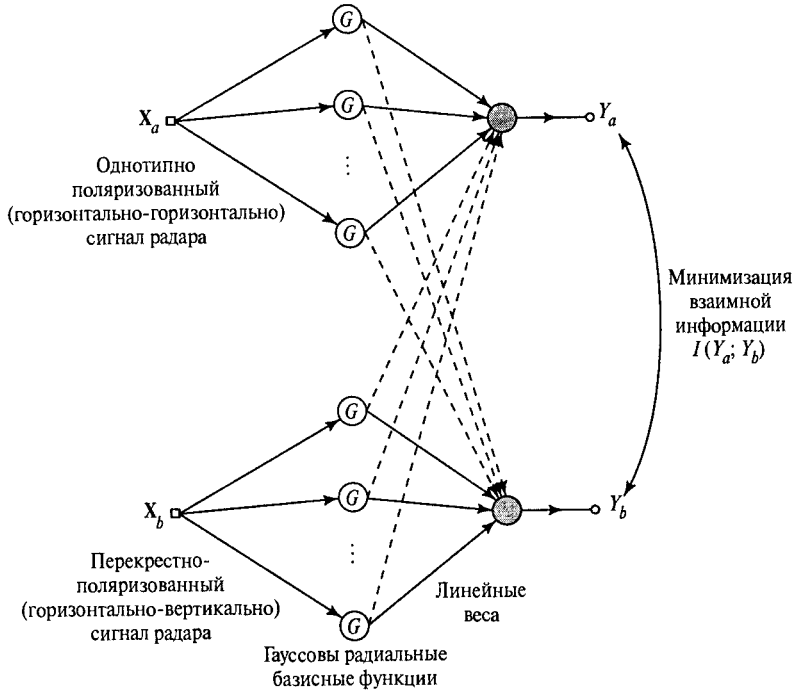


Рис. 10.7. Блочная диаграмма нейронного процессора, целью которого является подавление фонового шума с помощью пары несвязных радарных входов. Подавление помех производится за счет минимизации взаимной информации между выходами двух моделей

n обозначает дискретное время. Вектор $\mathbf{U}$ применяется к линейной системе, характеристики отображения входа на выход которой задаются несингулярной матрицей $\mathbf{A}$ размерности $m \times m$, называемой *матрицей смешения* (mixing matrix). Результатом является вектор наблюдений $\mathbf{X}(n)$ размерности $m \times 1$, связанный с вектором $\mathbf{U}$ следующим соотношением (см. рис. 10.10, *а*):

$$\mathbf{X} = \mathbf{AU}, \tag{10.72}$$

где

$$\mathbf{X} = [X_1, X_2, \ldots, X_m]^T.$$

Вектор источника $\mathbf{U}$ и матрица смешения $\mathbf{A}$ считаются неизвестными. Известен только вектор наблюдений $\mathbf{X}$. Для данного вектора $\mathbf{X}$ задача сводится к поиску *разделяющей* (demixing) *матрицы* $\mathbf{W}$, такой, что исходный вектор $\mathbf{U}$ может быть получен из выходного вектора $\mathbf{Y}$, определяемого следующим образом (см. рис. 10.10, *б*):

$$\mathbf{Y} = \mathbf{WX}, \tag{10.73}$$

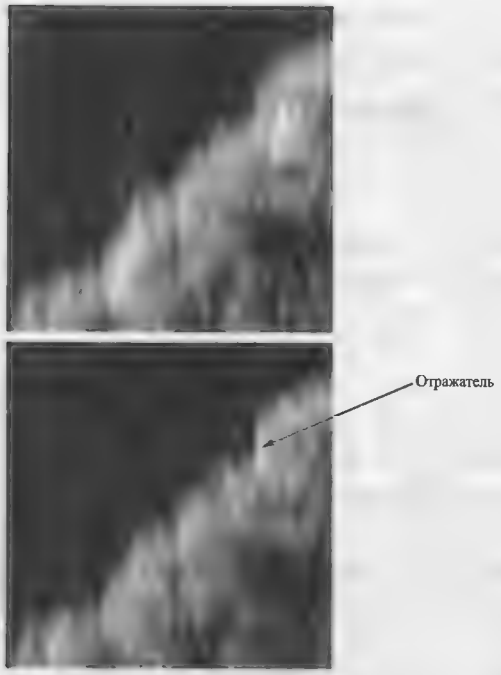


Рис.10.8,*а*. Рисунок, представляющий исходное радарное сканирование (азимут относительно диапазона) для горизонтально-горизонтальной (сверху) и горизонтально-вертикальной (снизу) поляризации

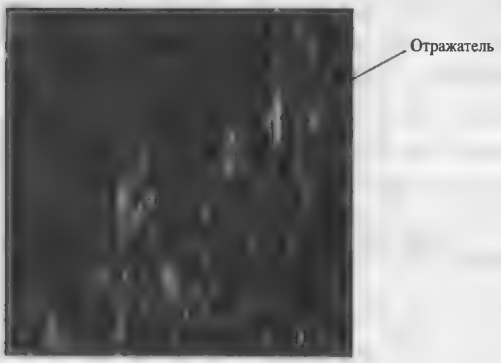


Рис.10.8,*б*. Композитный рисунок, вычисленный с помощью минимизации взаимной информации между двумя поляризованными изображениями, показанными на рис. 10.8, *а*

где

$$\mathbf{Y} = [Y_1, Y_2, \ldots, Y_m]^T.$$

Обычно в этой задаче предполагается, что средним значением входных сигналов $U_1, U_2, \ldots, U_m$ является нуль, что, в свою очередь, гарантирует нулевое среднее для элементов вектора наблюдений $X_1, X_2, \ldots, X_m$. То же можно сказать и об элементах выходного вектора разделяющей матрицы $Y_1, Y_2, \ldots, Y_m$.

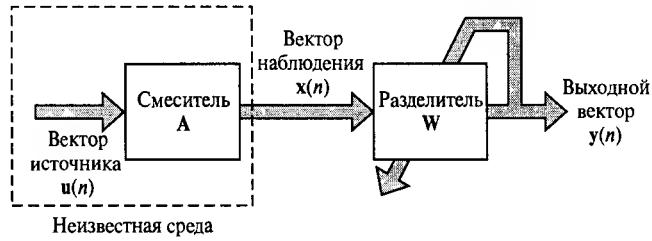


Рис. 10.9. Блочная диаграмма процессора для задачи слепого разделения источников. Векторы **u**, **x** и **y** являются соответственно значениями случайных векторов **U**, **X** и **Y**

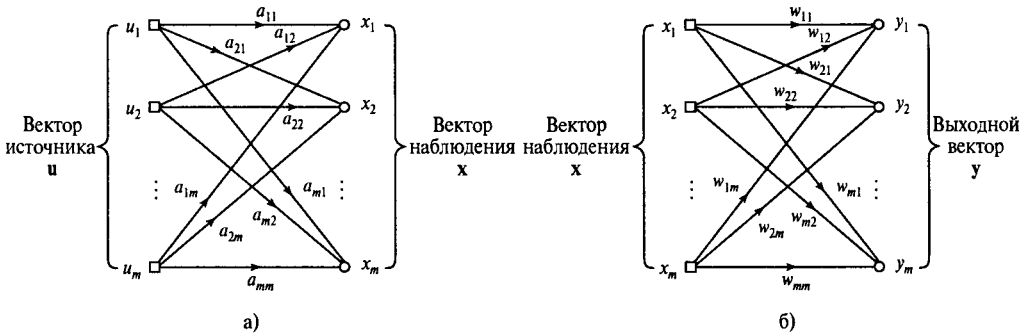


Рис. 10.10. Детализированное описание матрицы смешения (а) и разделяющей матрицы (б)

Таким образом, *задачу слепого разделения источников* (blind source separation problem) можно сформулировать следующим образом.

Для данных N независимых реализаций вектора наблюдений $\mathbf{X}$ *найти оценку матрицы, обратной матрице смешения* $\mathbf{A}$.

При разделении источников используется принцип *пространственного разнесения* (spatial diversity). Это проявляется в том, что разные сенсоры, являющиеся реализациями вектора $\mathbf{X}$, обслуживают разные смеси источников. Можно также использовать спектральное разнесение, если таковое существует. Однако фундаментальным подходом к разделению источников является *пространственный* — мы рассматриваем структуру в разрезе сенсоров, а не в разрезе времени [169].

Решение задачи слепого разделения источников достижимо, за исключением учета произвольного масштабирования каждого из компонентов сигнала и смешения индексов. Другими словами, вполне возможно найти такую разделяющую матрицу $\mathbf{W}$, отдельные строки которой являются масштабированными и перемешанными строками матрицы $\mathbf{A}$. Это значит, что решение можно представить в следующем виде:

$$\mathbf{Y} = \mathbf{WX} = \mathbf{WAU} \rightarrow \mathbf{DPU},$$

где $\mathbf{D}$ — несингулярная диагональная матрица; $\mathbf{P}$ — *матрица перемешивания* (permutation matrix).

Описанную выше задачу обычно называют задачей слепого разделения источников (сигналов)[13], где за термином "слепое" скрывается тот факт, что единственной информацией, используемой для восстановления входного сигнала, является реализация вектора наблюдений **X**, обозначаемая символом **x**. Принцип, на котором основано решение этой задачи, называется *анализом независимых компонентов* (independent component analysis — ICA) [205], который можно рассматривать как *расширение анализа главных* компонентов (principal components analysis — PCA). В то время как PCA предполагает независимость вплоть до второго порядка в требовании ортогональности векторов направлений, метод ICA требует *статистической независимости* отдельных компонентов выходного вектора **Y** и не связан с требованием ортогональности. Обратите внимание, что на практике алгоритмическая реализация анализа независимых компонентов может иметь место для "настолько статистически независимых компонентов, насколько это возможно".

Потребность в слепом разделении источников возникает в различных областях, в том числе следующих.

- *Разделение речи.* В этом приложении вектор **x** состоит из нескольких речевых сигналов, которые линейно смешаны друг с другом. При этом требуется разделить эти сигналы [116]. Сложная форма такой ситуации возникает, например, при проведении телеконференций.
- *Обработка антенной решетки* (array antenna processing). В этом приложении вектор **x** представляет собой выход антенной решетки радара, полученный из нескольких узкополосных сигналов, возникающих из источников, расположенных в неизвестных направлениях [174], [1038]. Здесь также требуется разделить источники сигналов. (Под узкополосным сигналом понимается полезный сигнал, имеющий меньшую полосу, по сравнению с несущей частотой.)
- *Мультисенсорные биомедицинские записи.* В этом приложении вектор **x** состоит из записей, выполненных множеством сенсоров, используемых для мониторинга биологических сигналов. Например, может потребоваться отделить сердцебиение беременной женщины от сердцебиения плода [170].
- *Анализ данных финансового рынка.* В этом приложении вектор **x** состоит из множества данных финансового рынка, и из него требуется извлечь доминантные независимые компоненты [84].

В этих приложениях задача слепого разделения источников может быть усложнена возможным присутствием неизвестных задержек распространения, расширенной

[13] Слепое разделение сигнала можно обнаружить еще в основополагающей работе [452]. Исторические сведения о задаче слепого разделения сигнала содержатся в [766]. Эта работа также освещает нейробиологические аспекты задачи. Глубокое исследование задачи слепого разделения источников, подчеркивающее заложенные в ней принципы обработки сигналов, содержится в [169].

фильтрацией, налагаемой на источники их средой, и неизбежным искажением вектора наблюдений $\mathbf{x}$ внешним шумом. Все это значит, что идеализированная форма смешения сигналов, описываемая формулой (10.72), очень редко встречается в реальных задачах. Однако это также значит и то, что для рассмотрения фундаментальных аспектов задачи разделения источников эти наложения следует игнорировать.

Критерий статистической независимости

Так как статистическая независимость является свойством, требуемым от компонентов выходного вектора $\mathbf{Y}$, возникает вопрос, какую меру использовать для ее определения? Как вариант можно выбрать взаимную информацию $I(Y_i;Y_j)$ между случайными переменными Y_i и Y_j, являющимися двумя компонентами выходного вектора $\mathbf{Y}$. Если (в идеальном случае) эта взаимная информация равна нулю, соответствующие компоненты Y_i и Y_j являются статистически независимыми. Этого можно добиться, минимизируя взаимную информацию между всеми парами случайных переменных, составляющих выходной вектор $\mathbf{Y}$. Эта цель эквивалентна минимизации дивергенции Кулбека–Лейблера между двумя следующими распределениями: функцией плотности вероятности $f_{\mathbf{Y}}(\mathbf{y},\mathbf{W})$, параметризованной по $\mathbf{W}$, и соответствующим факториальным распределением:

$$\tilde{f}_{\mathbf{Y}}(\mathbf{y},\mathbf{W}) = \prod_{i=1}^{m} \tilde{f}_{Y_i}(y_i,\mathbf{W}), \tag{10.74}$$

где $\tilde{f}_{\mathbf{Y}_i}(\mathbf{y}_i,\mathbf{W})$ — граничная функция плотности вероятности Y_i. В результате выражение (10.74) можно рассматривать как одно из ограничений, налагаемых на алгоритм обучения, направленное на противопоставление $f_Y(\mathbf{y},\mathbf{W})$ факториальному распределению $\tilde{f}_{\mathbf{Y}}(\mathbf{y},\mathbf{W})$. Таким образом, можно сформулировать третий вариант принципа Infomax следующим образом [205].

Для данного вектора $\mathbf{X}$ размерности $m \times 1$, представляющего линейную комбинацию m независимых источников сигнала, построить такое преобразование этого вектора наблюдения нейронной системой в новый вектор $\mathbf{Y}$, чтобы минимизировать дивергенцию Кулбека–Лейблера между параметризованной вероятностью, задаваемой функцией $f_{\mathbf{Y}}(\mathbf{y},\mathbf{W})$, и соответствующим факториальным распределением $\tilde{f}_{\mathbf{Y}}(\mathbf{y},\mathbf{W})$ по отношению к неизвестной матрице параметров $\mathbf{W}$.

Дивергенция Кулбека–Лейблера в описанной здесь задаче была представлена в разделе 10.5 формулой (10.44). Адаптируя эту формулу к текущей ситуации, дивергенцию Кулбека–Лейблера между функциями распределения вероятности $f_Y(\mathbf{y},\mathbf{W})$ и $\tilde{f}_{\mathbf{Y}}(\mathbf{y},\mathbf{W})$ можно записать следующим образом:

$$D_{f||\tilde{f}}(\mathbf{W}) = -h(\mathbf{Y}) + \sum_{i=1}^{m} \tilde{h}(Y_i), \tag{10.75}$$

где $h(\mathbf{Y})$ — энтропия случайного вектора $\mathbf{Y}$ на выходе разделителя сигнала; $\tilde{h}(Y_i)$ – граничная энтропия i-го элемента этого вектора. Дивергенция Кулбека–Лейблера $D_{f||\tilde{f}}$ является целевой функцией, на которой мы в дальнейшем и сконцентрируем внимание для решения задачи слепого разделения источников.

Определение дифференциальной энтропии $h(Y)$

Выходной вектор $\mathbf{Y}$ связан со входным вектором $\mathbf{X}$ соотношением (10.73), в котором $\mathbf{W}$ является разделяющей матрицей. В свете выражения (10.18) дифференциальную энтропию $\mathbf{Y}$ можно выразить следующим образом:

$$h(\mathbf{Y}) = h(\mathbf{WX}) = h(\mathbf{X}) + \log|\det(\mathbf{W})|, \tag{10.76}$$

где $\det(\mathbf{W})$ — определитель матрицы $\mathbf{W}$.

Определение граничной энтропии $\tilde{h}(Y_i)$

Для определения дивергенции Кулбека–Лейблера $D_{f||\tilde{f}}$ требуется также вычислить граничную энтропию $\tilde{h}(Y_i)$. Для этого необходимо знание граничного распределения переменной Y_i, что, в свою очередь, требует интегрирования эффекта от всех компонентов случайного вектора $\mathbf{Y}$, за исключением его i-го компонента. Если вектор $\mathbf{Y}$ имеет достаточно большую размерность, вычислить $\tilde{h}(Y_i)$ намного сложнее, чем $h(\mathbf{Y})$. Этой сложности можно избежать, если вывести приближенную формулу вычисления $\tilde{h}(Y_i)$ в терминах моментов высокого порядка случайной переменной Y_i. Этого можно добиться усечением одного из следующих разложений.

- Разложение в ряд Эджворса (Edgeworth) [206].
- Разложение в ряд Грама–Шарльера (Gram-Charlier) [37].

В этой главе мы рассмотрим второй подход[14].

---

[14] **Аппроксимация функции плотности вероятности**
(a) Разложение в ряд Грама–Шарльера (Gram-Charlier)
Пусть $\varphi_Y(\omega)$ — характеристическая функция$_Y(y)$. По определению

$$\varphi_Y(\omega) = \int_{-\infty}^{\infty} f_Y(y) e^{j\omega y} dy, \tag{1}$$

где $j = \sqrt{-1}$; ω — положительное число. Образно говоря, характеристическая функция $\varphi_Y(\omega)$ является преобразованием Фурье функции плотности вероятности $f_Y(y)$, за исключением перемены знака в экспоненте. В общем случае характеристическая функция $\varphi_Y(\omega)$ является комплексным числом, действительная и мнимая часть которого конечны для всех ω. Если существует k-й момент случайной переменной Y, то функция $\varphi_Y(\omega)$ может быть разложена в ряд в окрестности точки $\omega = 0$:

Для примера, разложение Грама-Шарльера (Gram-Charlier) параметризованной граничной функции плотности вероятности $\tilde{f}_{Y_i}(y_i, \mathbf{W})$ будет иметь следующий вид:

$$\tilde{f}_{Y_i}(y_i, \mathbf{W}) = \alpha(y_i)\left[1 + \sum_{k=3}^{\infty} c_k H_k(y_i)\right], \tag{10.77}$$

где используются следующие обозначения.

$$\varphi_Y(\omega) = 1 + \sum_{k=1}^{\infty} \frac{(j\omega)^k}{k!} m_k, \tag{2}$$

где m_k — момент k-го порядка случайной переменной Y:

$$m_k = E[Y^k] = \int_{-\infty}^{\infty} y^k f_Y(y) dy. \tag{3}$$

Выражение (2) было получено с помощью простой подстановки разложения экспоненциальной функции $e^{j\omega y}$ из (1), перемены порядка суммирования и интегрирования и применения определения (3). Если характеристическая функция $\varphi_Y(\omega)$ может быть разложена в ряд (2), тогда можно разложить в ряд и ее логарифм:

$$\log \varphi_Y(\omega) = \sum_{n=1}^{\infty} \frac{\kappa_n}{n!} (j\omega)^n, \tag{4}$$ где κ_n

называется накоплением n-го порядка (cumulant) *или* псевдоинвариантом (semi-invariant) случайной переменной Y. Равенство (4) было получено разложением логарифма функции $\varphi_Y(\omega)$ в ряд Тейлора по $j\omega$ в окрестности точки $\omega = 0$.

Для упрощения изложения сделаем два допущения.

Случайная переменная Y имеет нулевое среднее, т.е. $\mu = 0$.

Дисперсия Y нормализована до единицы, т.е. $\sigma^2 = 1$.

Следовательно, $\kappa_1 = 0 \kappa_2 = 1$, и равенство (4) принимает следующий вид:

$$\log \varphi_Y(\omega) = \frac{1}{2}(j\omega)^2 + \sum_{n=3}^{\infty} \frac{\kappa_n}{n!} (j\omega)^n. \tag{5}$$

Пусть $r(\omega) = \sum_{n=3}^{\infty} \frac{\kappa_n}{n!} (j\omega)^n$.

Тогда равенство (5) можно переписать в следующем виде: $\log \varphi_Y(\omega) = \frac{1}{2}(j\omega)^2 + r(\omega)$.

Это значит, что характеристическая функция $\varphi_Y(\omega)$ может быть представлена как произведение двух экспонент:

$$\varphi_Y(\omega) = \exp\left(-\frac{\omega^2}{2}\right) \cdot \exp(r(\omega)). \tag{6}$$

Используя разложение степенного ряда для слагаемого $\exp(r(\omega))$, получим:

$$\exp(r(\omega)) = 1 + \sum_{l=1}^{\infty} \frac{r^l(\omega)}{l!}. \tag{7}$$

Подставляя (7) в (6) и собирая слагаемые с одинаковыми степенями $(j\omega)$ в результате двойного суммирования, получим новые коэффициенты разложения функции $\varphi_Y(\omega)$:

$$c_1 = 0, \quad c_2 = 0,$$
$$c_3 = \frac{\kappa_3}{6}, \quad c_4 = \frac{\kappa_4}{24}, \quad c_5 = \frac{\kappa_5}{120},$$
$$c_6 = \frac{1}{720}(\kappa_6 + 10\kappa_3^2), \quad c_7 = \frac{1}{5040}(\kappa_7 + 35\kappa_4\kappa_3), \quad c_8 = \frac{1}{40320}(\kappa_8 + 56\kappa_5\kappa_3 + 35\kappa_4^2)$$

и т.д. Теперь можно выполнить обратное преобразование Фурье функции $\varphi_Y(\omega)$ и получить разложение функции плотности вероятности $f_Y(y)$. В частности, можно записать:

$$f_Y(y) = \alpha(y)\left(1 + \sum_{k=3}^{\infty} c_k H_k(y)\right), \tag{8}$$

где $\alpha(y)$ — функция плотности вероятности *нормализованной гауссовой случайной переменной* с нулевым средним и единичной дисперсией:

$$\alpha(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}. \tag{9}$$

1. Масштабирующий множитель $\alpha(y_i)$ является функцией плотности вероятности нормализованной гауссовой случайной переменной с нулевым средним и единичной дисперсией, т.е.
$$\alpha(y_i) = \frac{1}{\sqrt{2\pi}} e^{-y_i^2/2}.$$
2. $H_k(y_i)$ — *полиномы Эрмита* (Hermite polynomial).
3. Коэффициенты разложения $\{c_k\colon k = 3,4,\dots\}$ определены в терминах *семиинвариантов* (cumulants) случайной переменной Y_i.

Разложение (8) называется рядом Грама–Шарльера (Gram-Charlier) функции плотности вероятности по функцням Гаусса и их производным [1022]. Разложение такого рода имеет интуитивную привлекательность. В частности, если случайная переменная Y состоит из суммы нескольких независимо и равномерно распределенных случайных переменных, тогда по мере увеличения колнчества этих переменных, согласно центральной предельной теореме, такая случайная переменная Y асимптотически становится гауссовой. Первое слагаемое ряда Грама–Шарльера является гауссовым. Это значит, что сумма оставшихся слагаемых ряда стремится к нулю по мере увеличения количества переменных в сумме. Полиномы Эрмита $H_k(y)$, которые используются в разложении (8), определяются в терминах k-х производных $\alpha(y)$ следующим образом:
$$\alpha^{(k)}(y) = (-1)^k \alpha(y) H_k(y). \tag{10}$$
Вот некоторые типичные полиномы Эрмита:
$H_0(y) = 1,$
$H_1(y) = y,$
$H_2(y) = y^2 - 1,$
$H_3(y) = y^3 - 3y,$
$H_4(y) = y^4 - 6y^2 + 3,$
$H_5(y) = y^5 - 10y^3 + 15y,$
$H_6(y) = y^6 - 15y^4 + 45y^2 - 15.$
Рекурсивное выражение для вычисления этих полиномов имеет следующий вид:
$$H_{k+1}(y) = yH_k(y) - kH_{k-1}(y). \tag{11}$$
Особенно важным свойством полиномов Эрмита является то, что $H_k(y)$ и m-я производная функции Гаусса $\alpha(y)$ являются биортогональными (biorthogonal):
$$\int_{-\infty}^{\infty} H_k(y)\alpha^{(m)}(y)dy = (-1)^m m!\delta_{km}, \quad (k,m) = 0,1,\dots. \tag{12}$$
Величина δ_{km} называется дельта-функцией Кронекера (Kronecker delta), которая принимает значение 1, если $k = m$, и 0 — в противном случае.

Важно заметить, что в разложенин в ряд Грама–Шарльера (Gram-Charlier) естественный порядок слагаемых не является лучшим. Вместо этого целесообразнее сгруппировать слагаемые следующим образом [449]:
$$k = (0),\ (3),\ (4,6),\ (5,7,9). \tag{13}$$
Элементы в группах обычно имеют один порядок амплитуды. Например, если включить слагаемое с $k = 4$, то необходимо также включить и слагаемое с $k = 6$.

(б) Разложенне в ряд Эджворса (Edgeworth)

Как и в первом случае, пусть $\alpha(y)$ — функция плотности вероятности некоторой нормализованной случайной переменной с нулевым средним значением и едничной дисперсией. Разложение в ряд Эджворса функции плотности вероятности случайной переменной Y в окрестности гауссовой аппроксимации $\alpha(y)$ определяется следующей формулой [205], [1023]:
$$\begin{aligned}\frac{f_Y(y)}{\alpha(y)} &= 1 + \frac{\kappa_3}{3!}H_3(y) + \frac{\kappa_4}{4!}H_4(y) + \frac{10\kappa_3^2}{6!}H_6(y) + \frac{\kappa_5}{5!}H_5(y) + \\ &+ \frac{35\kappa_3\kappa_4}{7!}H_7(y) + \frac{280\kappa_3^3}{9!}H_9(y) + \frac{\kappa_5}{6!}H_6(y) + \frac{56\kappa_3\kappa_5}{8!}H_8(y) + \\ &+ \frac{35\kappa_4^2}{8!}H_8(y) + \frac{2100\kappa_3^2\kappa_4}{10!}H_{10}(y) + \frac{15400\kappa_3^4}{12!}H_{12}(y) + \dots,\end{aligned} \tag{14}$$
где κ_i — накопление порядка i стандартизированной скалярной случайной переменной Y; H_i — полином Эрмита порядка i. Разложение (14) называется рядом Эджворса

Ключевым свойством ряда Эджворса является то, что его коэффициенты уменьшаются равномерно. С другой стороны, слагаемые ряда Грама–Шарльера (8) не сходятся равномерно к нулю, поэтому нельзя сказать, что какое-либо слагаемое привносит меньший вклад, чем предыдущее. Именно по этой причине при обрезании этого ряда рекомендована процедура группировки его слагаемых (13).

Естественный порядок слагаемых в выражении (10.77) *не* является лучшим для разложения Грама–Шарльера (Gram-Charlier). В этом разложении слагаемые лучше сгруппировать следующим образом [449]:

$$k = (0), (3), (4,6), (5,7,9), \ldots .$$

Для задачи слепого разделения источников аппроксимацию граничной функции плотности вероятности $\tilde{f}_{Y_i}(y_i)$ путем усечения разложения Грама–Шарльера на уровне k =(4,6) можно считать адекватным. Исходя из этого, можно записать:

$$\tilde{f}_{Y_i}(y_i) \simeq \alpha(y_i)\left(1 + \frac{\kappa_{i,3}}{3!}H_3(y_i) + \frac{\kappa_{i,2}^2}{4!}H_4(y_i) + \frac{(\kappa_{i,6} + 10\kappa_{i,3}^2)}{6!}H_6(y_i)\right), \quad (10.78)$$

где $\kappa_{i,k}$ — *семиинвариант* (cumulant) k-го порядка переменной Y_i. Обозначим как $m_{i,k}$ момент k-го порядка переменной Y_i:

$$m_{i,k} = E[Y_i^k] = E\left[\left(\sum_{k=1}^{m} w_{ik}X_i\right)^k\right], \quad (10.79)$$

где X_i — i-й элемент вектора наблюдений $\mathbf{X}$; w_{ik} — ik-й элемент матрицы весов $\mathbf{W}$. Ранее уже было обосновано предположение о нулевом среднем компонентов Y_i для всех i. Следовательно, $\sigma_i^2 = m_{i,2}$ (т.е. дисперсия равна среднеквадратическому значению). Исходя из этого, семиинварианты переменной Y_i можно связать с ее моментами следующим образом:

$$\kappa_{i,3} = m_{i,3}, \quad (10.80)$$

$$\kappa_{i,4} = m_{i,4} - 3m_{i,2}^2, \quad (10.81)$$

$$\kappa_{i,6} = m_{i,6} - 10m_{i,3}^2 - 15m_{i,2}m_{i,4} + 30m_{i,2}^3. \quad (10.82)$$

Алгоритм нахождения $\tilde{f}_{Y_i}(y_i)$ с помощью аппроксимации (10.78) можно представить так:

$$\log \tilde{f}_{Y_i}(y_i) \simeq \log \alpha(y_i) + \\ + \log\left(1 + \frac{\kappa_{i,3}}{3!}H_3(y_i) + \frac{\kappa_{i,2}^2}{4!}H_4(y_i) + \frac{(\kappa_{i,6} + 10\kappa_{i,3}^2)}{6!}H_6(y_i)\right). \quad (10.83)$$

Далее будем использовать следующее разложение логарифма в ряд:

$$\log(1+y) \simeq y - \frac{y^2}{2}, \quad (10.84)$$

где все слагаемые третьего порядка и выше игнорируются.

Вспомним из предыдущего материала, что формула граничной энтропии Y_i имеет следующий вид (см. (10.43)):

$$\tilde{h}(Y_i) = -\int_{-\infty}^{\infty} f_{Y_i}(y_i)\log f_{Y_i}(y_i)dy_i, \quad i = 1, 2, \ldots, m,$$

где m — количество источников. Используя аппроксимации (10.78), (10.83) и (10.84) и взяв определенные интегралы, содержащие нормализованные гауссовы плотности $\alpha(y_i)$ и различные полиномы Эрмита $H_k(y_i)$, получим следующую приближенную формулу для граничной энтропии [700]:

$$\begin{aligned}\tilde{h}(Y_i) \simeq \frac{1}{2}\log(2\pi e) - \frac{\kappa_{i,3}^2}{12} - \frac{\kappa_{i,4}^2}{48} - \frac{(\kappa_{i,6}+10\kappa_{i,3}^2)^2}{1440} + \frac{3}{8}\kappa_{i,3}^2\kappa_{i,4} + \\ + \frac{\kappa_{i,3}^2(\kappa_{i,6}+10\kappa_{i,3}^2)}{24} + \frac{\kappa_{i,4}^2(\kappa_{i,6}+10\kappa_{i,3}^2)}{24} + \frac{\kappa_{i,4}^(\kappa_{i,6}+10\kappa_{i,3}^2)^2}{64} + \\ + \frac{\kappa_{i,4}^3}{16} + \frac{(\kappa_{i,6}+10\kappa_{i,3}^2)^3}{432}.\end{aligned} \tag{10.85}$$

Подставляя (10.76) и (10.85) в (10.75), получим выражение для дивергенции Кулбека–Лейблера поставленной задачи:

$$\begin{aligned}D_{f||\hat{f}}(\mathbf{W}) \simeq -h(\mathbf{X}) - \log|\det(\mathbf{W})| + \frac{m}{2}\log(2\pi e) - \\ - \sum_{i=1}^{m}(\frac{\kappa_{i,3}^2}{12} + \frac{\kappa_{i,4}^2}{48} + \frac{(\kappa_{i,6}+10\kappa_{i,3}^2)^2}{1440} - \frac{3}{8}\kappa_{i,3}^2\kappa_{i,4} - \\ - \frac{\kappa_{i,3}^2(\kappa_{i,6}+10\kappa_{i,3}^2)}{24} - \frac{\kappa_{i,4}^2(\kappa_{i,6}+10\kappa_{i,3}^2)}{24} - \frac{\kappa_{i,4}^(\kappa_{i,6}+10\kappa_{i,3}^2)^2}{64} - \\ - \frac{\kappa_{i,4}^3}{16} - \frac{(\kappa_{i,6}+10\kappa_{i,3}^2)^3}{432}),\end{aligned} \tag{10.86}$$

где все семиинварианты являются функциями матрицы весов $\mathbf{W}$.

Функция активации

Чтобы оценить дивергенцию Кулбека–Лейблера, описанную формулой (10.86), требуется адаптивная процедура вычисления семиинвариантов высокого порядка вектора наблюдений $\mathbf{x}$. Вопрос состоит в следующем: как выполнить эти вычисления, учитывая способ вывода приближенной формулы (10.86)? Вспомним, что приведенный вывод этой формулы основывался на разложении Грама–Шарльера, при этом предполагалось, что случайная переменная Y_i имеет нулевое среднее и единичную дисперсию. Ранее мы уже обосновывали нулевое среднее тем, что для начала входные

сигналы будут иметь нулевое среднее. Что же касается предположения о единичной дисперсии, для его обоснования можно применить один из следующих подходов.

1. *Подход с ограничениями.* В этом подходе при вычислении *семиинвариантов* высокого порядка $\kappa_{i,3}, \kappa_{i,4}$ и $\kappa_{i,6}$ для всех i делается предположение о единичной дисперсии [37]. К сожалению, нет никакой гарантии, что на протяжении вычислений дисперсия Y_i, а именно σ_i^2, останется константой, равной единице. Из определений (10.81) и (10.82) видно, что как $\kappa_{i,4}$, так и $\kappa_{i,6}$ зависят от $\sigma_i^2 = m_{i,2}$. В результате предположение о равенстве дисперсии единице сводится к тому, что оценки, производные от $\kappa_{i,4}$ и $\kappa_{i,6}$, смещаются и становятся ошибочно связанными с оценкой $\kappa_{i,3}$.
2. *Подход без ограничений.* В этом альтернативном подходе дисперсия σ_i^2 рассматривается как неизвестный параметр, зависящий от времени, что весьма близко к реальности [700]. Эффект от отклонения дисперсии от единицы рассматривается как масштабирующая вариация значений случайной переменной Y_i. Что более важно — оценки, производные от $\kappa_{i,4}$ и $\kappa_{i,6}$, учитывают изменение σ_i^2 во времени. Таким образом, формируется правильное соотношение между оценками всех трех семиинвариантов в формуле (10.86).

Экспериментальное изучение слепого разделения источников было проведено в [700]. В этой работе показано, что подход без ограничений приводит к более высокой производительности, чем подход с ограничениями. В связи с этим в дальнейшем будем использовать именно его (подход без ограничений).

Чтобы построить алгоритм обучения для вычисления $\mathbf{W}$, необходимо продифференцировать выражение (10.86) по $\mathbf{W}$, а затем вывести функцию активации алгоритма.

Пусть A_{ik} — ik-й *кофактор* (cofactor) матрицы $\mathbf{W}$. Используя *разложение Лапласа* определителя $\det(\mathbf{W})$ по i-й строке, можно записать:

$$\det(\mathbf{W}) = \sum_{k=1}^{m} w_{ik} A_{ik}, \; i = 1, 2, \ldots, m, \tag{10.87}$$

где w_{ik} — ik-й элемент матрицы $\mathbf{W}$. Исходя из этого, дифференцируя логарифм определителя $\det(\mathbf{W})$ по w_{ik}, получим:

$$\frac{\partial}{\partial w_{ik}} \log(\det(\mathbf{W})) = \frac{1}{\det(\mathbf{W})} \frac{\partial}{\partial w_{ik}} \det(\mathbf{W}) = \frac{A_{ik}}{\det(\mathbf{W})} = (\mathbf{W}^{-T})_{ik}, \tag{10.88}$$

где $\mathbf{W}^{-T}$ — матрица, обратная транспонированной матрице $\mathbf{W}^T$. Частные производные остальных слагаемых (зависящих от $\mathbf{W}$) в выражении (10.86) по w_{ik} равны (см. (10.80)–(10.82)):

$$\frac{\partial \kappa_{i,3}}{\partial w_{ik}} = 3E[Y_i^2 X_k],$$
$$\frac{\partial \kappa_{i,4}}{\partial w_{ik}} = 4E[Y_i^3 X_k] - 12m_{i,2}E[Y_i X_k],$$
$$\frac{\partial}{\partial w_{ik}}(\kappa_{i,6} + 10\kappa_{i,3}^2) = 6E[Y_i^5 X_k] - 30m_{i,4}E[Y_i X_k] - 60m_{i,2}E[Y_i^3 X_k] + 180m_{i,2}^2 E[Y_i X_k].$$

При выводе адаптивного алгоритма обычно значения математического ожидания заменяются их моментальными значениями. Таким образом, производя такую замену во всех трех уравнениях, получим следующие приближенные результаты:

$$\frac{\partial \kappa_{i,3}}{\partial w_{ik}} \simeq 3y_i^2 x_k, \tag{10.89}$$
$$\frac{\partial \kappa_{i,4}}{\partial w_{ik}} \simeq -8y_i^3 x_k, \tag{10.90}$$
$$\frac{\partial}{\partial w_{ik}}(\kappa_{i,6} + 10\kappa_{i,3}^2) \simeq 96y_i^5 x_k. \tag{10.91}$$

Подставляя (10.88)–(10.91) в выражение для производной (10.86) по w_{ik}, получим:

$$\frac{\partial}{\partial w_{ik}} D_{f||\tilde{f}}(\mathbf{W}) \simeq (\mathbf{W}^{-T})_{ik} + \varphi(y_i)x_k, \tag{10.92}$$

где $\varphi(y_i)$ — немонотонная *функция активации* алгоритма обучения, определенная следующим образом [700]:

$$\varphi(y_i) = \frac{1}{2}y_i^5 + \frac{2}{3}y_i^7 + \frac{15}{2}y_i^9 + \frac{2}{15}y_i^{11} - \frac{112}{3}y_i^{13} + 128y_i^{15} - \frac{512}{3}y_i^{17}. \tag{10.93}$$

На рис. 10.11 представлен график функции активации $\varphi(y_i)$ для y_i, лежащих в открытом интервале (–1,+1). Это покрывает диапазон выходов y_i, с которыми обычно работает алгоритм обучения. Обратите внимание, что наклон функции активации положителен в интервале (–0.734, +0.734). Это — требование для устойчивости алгоритма, о котором речь пойдет ниже.

Алгоритм обучения для ICA

Целью алгоритма обучения является минимизация дивергенции Кулбека–Лейблера между функцией плотности вероятности $\mathbf{Y}$ и факториальным распределением $Y_i, i = 1, 2, \ldots, m$. Минимизации можно достичь с помощью метода градиентного спуска, в котором корректировке подвергается вес w_{ik}:

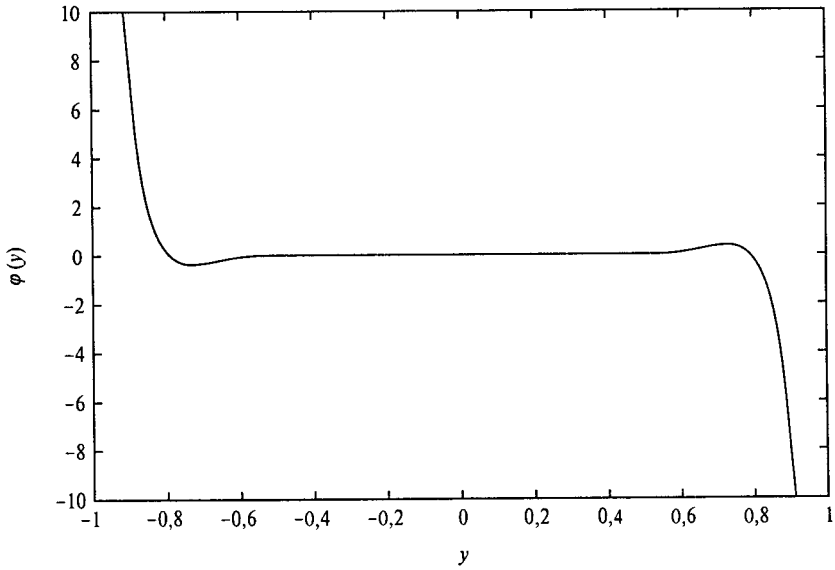


Рис. 10.11. Функция активации, задаваемая формулой (10.93)

$$\Delta w_{ik} = -\eta \frac{\partial}{\partial w_{ik}} D_{f||\tilde{f}} = \eta \left((\mathbf{W}^{-T})_{ik} - \varphi(y_i) x_k \right), \tag{10.94}$$

где η — параметр скорости обучения.

Расширяя формулу (10.94) на всю матрицу весов $\mathbf{W}$, корректировку $\Delta\mathbf{W}$, применяемую к вектору $\mathbf{W}$, можно выразить соотношением

$$\Delta\mathbf{W} = \eta(\mathbf{W}^{-T} - \varphi(\mathbf{y})\mathbf{x}^T), \tag{10.95}$$

где $\mathbf{x}^T$ — транспонированный вектор наблюдений, а

$$\boldsymbol{\varphi}(\mathbf{y}) = [\varphi(y_1), \varphi(y_2), \ldots, \varphi(y_m)]^T. \tag{10.96}$$

Формулу (10.95) можно переписать в следующем виде:

$$\Delta\mathbf{W} = \eta[\mathbf{I} - \varphi(\mathbf{y})\boldsymbol{x}^T\mathbf{W}^T]\mathbf{W}^{-T} = \eta[\mathbf{I} - \varphi(\mathbf{y})\mathbf{y}^T]\mathbf{W}^{-T}, \tag{10.97}$$

где $\mathbf{I}$ — единичная матрица. Правило коррекции при адаптации разделяющей матрицы примет следующий вид:

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \eta(n)[\mathbf{I} - \boldsymbol{\varphi}(\mathbf{y}(n))\mathbf{y}^T(n)]\mathbf{W}^{-T}(n). \tag{10.98}$$

В этой формуле все параметры показаны в форме, зависящей от времени.

Свойство эквивариантности

Целью алгоритма слепого разделения источников является такая коррекция матрицы разделения $\mathbf{W}(n)$, чтобы вектор

$$\mathbf{y}(n) = \mathbf{W}(n)\mathbf{x}(n) = \mathbf{W}(n)\mathbf{A}\mathbf{u}(n)$$

был как можно ближе к исходному вектору $\mathbf{u}(n)$ в некотором статистическом смысле. Для примера рассмотрим глобальную систему, характеризуемую матрицей $\mathbf{C}(n)$, которая получена перемножением матрицы смешения $\mathbf{A}$ и матрицы разделения $\mathbf{W}(n)$:

$$\mathbf{C}(n) = \mathbf{W}(n)\mathbf{A}. \tag{10.99}$$

В идеальном случае эта глобальная система будет удовлетворять двум условиям.

1. Алгоритм, отвечающий за корректировку $\mathbf{C}(n)$, сходится к оптимальному значению, равному матрице *перестановок* (permutation).
2. Сам алгоритм можно представить в следующем виде:

$$\mathbf{C}(n+1) = \mathbf{C}(n) + \eta(n)\mathbf{G}(\mathbf{C}(n)\mathbf{u}(n))\mathbf{C}(n), \tag{10.100}$$

где $\mathbf{G}(\mathbf{C}(n)\mathbf{u}(n))$ — вектор-функция аргумента $\mathbf{C}(n)\mathbf{u}(n)$. Производительность этого алгоритма в целом характеризуется системной матрицей $\mathbf{C}(n)$, а не индивидуальными значениями матриц смешения $\mathbf{A}$ и разделения $\mathbf{W}(n)$. Такая адаптивная система называется *эквивариантной* [173].

Можно показать, что адаптивный алгоритм (10.98) приближенно удовлетворяет первому условию. Для этого можно переписать (10.98) в эквивалентной форме:

$$\mathbf{C}(n+1) = \mathbf{C}(n) + \eta(n)\mathbf{G}(\mathbf{C}(n)\mathbf{u}(n))\mathbf{W}^{-T}(n)\mathbf{A}, \tag{10.101}$$

где

$$\mathbf{G}(\mathbf{C}(n)\mathbf{u}(n)) = \mathbf{I} - \boldsymbol{\varphi}(\mathbf{C}(n)\mathbf{u}(n))(\mathbf{C}(n)\mathbf{u}(n))^T. \tag{10.102}$$

Алгоритм (10.98) быстро переходит в эквивариантное состояние (10.100), если вектор-функцию $\mathbf{G}(\mathbf{C}(n)\mathbf{u}(n))$ домножить на $\mathbf{W}^{-T}\mathbf{A}$, что, собственно, является еще одной формой $\mathbf{C}(n)$. Эту ситуацию можно исправить, заменив это значение на матричное произведение $\mathbf{W}^T(n)\mathbf{W}(n)$. Слагаемое $\mathbf{W}^T\mathbf{W}$, составленное из произведения матрицы $\mathbf{W}$ на саму себя транспонированную, всегда является положительно опре-

деленным. Это и является причиной того, что перемножение на $\mathbf{W}^T\mathbf{W}$ не изменяет знака минимума алгоритма обучения.

Возникает важный вопрос: как применить эту модификацию, чтобы достичь эквивариантного состояния? Ответ на этот вопрос заключается в самой формулировке метода градиентного спуска в пространстве параметров. В идеальном случае можно использовать *натуральный градиент*[15] (natural gradient) целевой функции $D_{f||g}(\mathbf{W})$, определяемой в терминах обычного градиента $\nabla D_{f||\tilde{f}}(\mathbf{W})$ следующим образом:

$$\nabla^* D_{f||\tilde{f}}(\mathbf{W}) = (\nabla D_{f||\tilde{f}}(\mathbf{W}))\mathbf{W}^T\mathbf{W}. \tag{10.103}$$

Сама матрица обычного градиента $\nabla D_{f||\tilde{f}}(\mathbf{W})$ представляет собой оптимальное направление спуска только в случае, когда пространство параметров $\mathbf{W}=\{\mathbf{W}\}$ является Евклидовым и имеет ортонормированную систему координат. Однако в нейронных сетях при нормальных условиях система координат пространства параметров обычно не является ортонормированной. В последней ситуации *наискорейший спуск* (steepest descent) можно реализовать с помощью натурального градиента $\nabla^* D_{f||\tilde{f}}(\mathbf{W})$. Таким образом, его применение в стохастическом алгоритме для слепого разделения источников будет более предпочтительным, чем использование обычного градиента. Чтобы пространство натуральных градиентов было определимо, необходимо выполнение следующих условий.

1. Пространство параметров $\mathbf{W}$ должно быть *Римановым*[16]. Риманова структура является дифференцируемым множеством с положительно определенной метрикой $\mathbf{W}$.
2. Матрица $\mathbf{W}$ должна быть несингулярной (т.е. обратимой).

В рассматриваемой нами задаче оба эти условия выполняются.

Модифицируя алгоритм (10.98) описанным выше способом, можно записать:

$$\begin{aligned}\mathbf{W}(n+1) &= \mathbf{W}(n) + \eta(n)[\mathbf{I} - \boldsymbol{\varphi}(\mathbf{y}(n))\mathbf{y}^T(n)](\mathbf{W}(n)\mathbf{W}^T(n))\mathbf{W}^{-T}(n) = \\ &= \mathbf{W}(n) + \eta(n)[\mathbf{I} - \boldsymbol{\varphi}(\mathbf{y}(n))\mathbf{y}^T(n)]\mathbf{W}(n).\end{aligned} \tag{10.104}$$

---

[15] Идея использования величины $\nabla * D = (\nabla D)\mathbf{W}^T\mathbf{W}$ вместо обычного градиента ∇D для задачи разделения источников была впервые описана в [173]. В этой работе величина ∇*D получила название относительного градиента (relative gradient). Этот градиент в точности совпадает с *натуральным градиентом*

[16] В Римановом пространстве размерности n, например, квадратичная норма вектора **a** определяется следующим образом:

$$\|\mathbf{a}\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}a_i a_j,$$

где g_{ij} – функции координат $x_1, x_2, \ldots, x_n$ Риманова пространства, $g_{ij} = g_{ji}$, а правая часть этого выражения всегда положительна. Это выражение является обобщением формулы Евклида для квадратичной нормы:

$$\|\mathbf{a}\|^2 = \sum_{i=1}^{n} a_i^2.$$

Риманова структура рассматривается в [26], [763].

Это выражение приводит к слепому разделению источников со свойством эквивариантности. На рис. 10.12 показано представление формулы (10.104) в виде графа передачи сигнала.

Для того чтобы адаптивный алгоритм (10.104) давал корректное решение задачи слепого разделения источников (см. рис. 10.9), для всех компонентов выходного вектора $\mathbf{Y}$ должны выполняться следующие условия.

- Ряд Грама–Шарльера, используемый для вычисления нелинейной функции $\varphi(\cdot)$, должен содержать достаточное число слагаемых. Это обеспечит хорошую аппроксимацию граничной энтропии $h(Y_i)$. Например, этому требованию вполне удовлетворяет функция (10.93).
- Для достоверности оценки *семиинвариантов* Y_i параметр скорости обучения η должен быть достаточно мал.

Условия устойчивости

Разговор о задаче слепого разделения источников был бы не полным, если бы мы не рассмотрели вопросы устойчивости адаптивного алгоритма, описываемого формулой (10.104). В [36] такое исследование было проведено для произвольной функции активации $\varphi(\cdot)$. Этот анализ проводился в контексте асимптотической сходимости алгоритма к желаемой точке равновесия, в которой гарантировалось успешное разделение источников.

Выражение (10.104) является дискретным описанием алгоритма слепого разделения источников, основанного на натуральном градиенте. Для анализа устойчивости рассмотрим его непрерывный эквивалент:

$$\dot{\mathbf{W}}(t) = \eta(t)[\mathbf{I} - \varphi(\mathbf{y}(t))\mathbf{y}^T(t)]\mathbf{W}(t), \tag{10.105}$$

где t — непрерывная переменная времени; $\dot{\mathbf{W}}(t) = \partial\mathbf{W}(t)/\partial t$. Параметр скорости обучения $\eta(t)$ является положительным на всем временном интервале. Пусть

$$\sigma_i^2 = E[y_i^2], \tag{10.106}$$

$$k_i = E\left[\frac{\partial\varphi(y_i)}{\partial y_i}\right], \tag{10.107}$$

$$q_i = E\left[y_i^2\frac{\partial\varphi(y_i)}{\partial y_i}\right]. \tag{10.108}$$

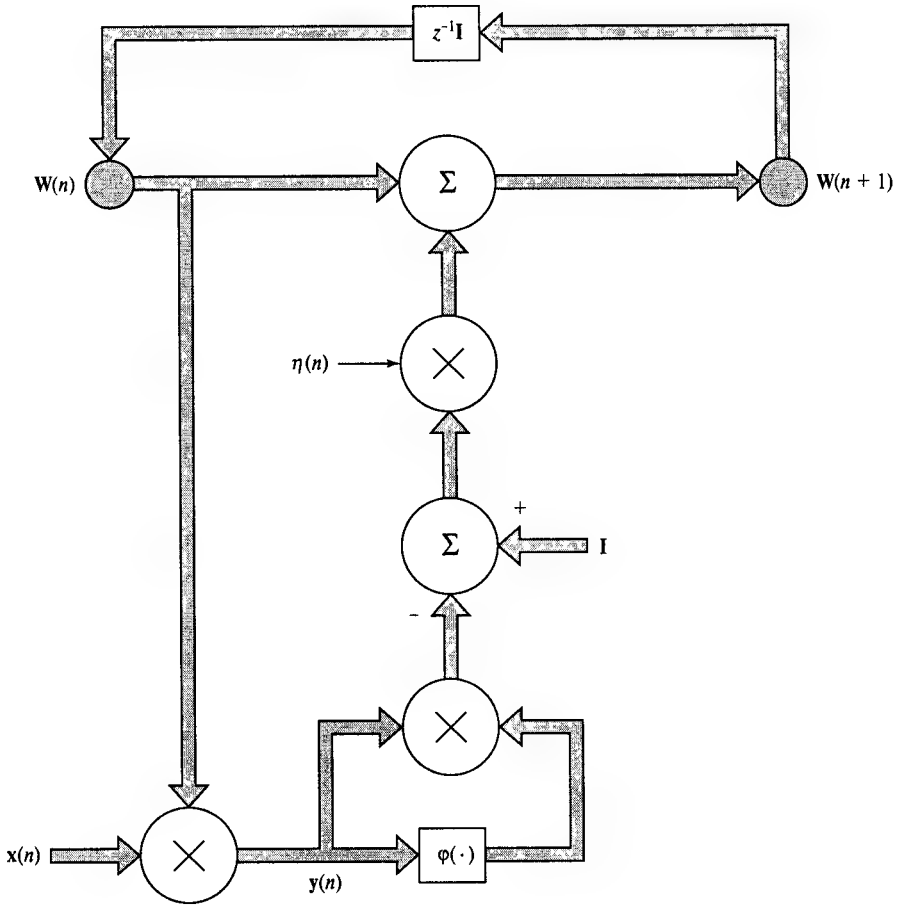


Рис. 10.12. Граф передачи сигнала для алгоритма обучения слепому разделению сигнала (10.104)

Тогда, согласно [36], разделяющее решение является устойчивой точкой равновесия адаптивного алгоритма (10.104) для произвольной функции активации $\varphi(\cdot)$ тогда и только тогда, когда выполняются следующие условия:

$$q_i + 1 > 0, \tag{10.109}$$

$$k_i > 0, \tag{10.110}$$

$$\sigma_i^2 \sigma_j^2 k_i k_j > 1. \tag{10.111}$$

для всех пар (i, j), где $i \neq j$. Неравенства (10.109)–(10.111) являются необходимым и достаточным условием устойчивости адаптивного алгоритма (10.104).

Условия сходимости

Что можно сказать о сходимости алгоритма обучения (10.104), основанного на функции активации (10.93), если выполнены требования устойчивости (10.109)–(10.111)? В свете экспериментальных исследований, изложенных в [700], можно утверждать, что процесс сходимости проходит две фазы.

- В первой фазе корректировке подвергается дисперсия $\sigma_i^2(n)$ случайной переменной Y_i на выходе разделителя, достигая в результате достаточно устойчивого значения. Во время этой фазы семиинварианты $\kappa_{i,3}$, $\kappa_{i,4}$ и $\kappa_{i,6}$ существенно не изменяются.
- Во второй фазе процесс корректировки распространяется на семиинварианты $\kappa_{i,3}$, $\kappa_{i,4}$ и $\kappa_{i,6}$, достигая в результате достаточно устойчивого значения. После этого можно считать, что алгоритм сошелся.

Таким образом, получается, что оценка дисперсии и *семиинвариантов* высоких порядков на выходе разделителя (т.е. разделенный сигнал источника) представляют собой базис чувствительной процедуры изучения сходимости алгоритма обучения (10.104). Интересно заметить также и то, что только во второй фазе используется разложение в ряд Грама–Шарльера.

10.12. Компьютерное моделирование

Рассмотрим систему, показанную на рис. 10.9 и имеющую следующие источники:

$u_1(n) = 0,1 \cdot \sin(400n)\cos(30n)$,
$u_2(n) = 0,1 \cdot \mathrm{sgn}(\sin(500n + 9\cos(40n)))$,
$u_2(n)$ соответствует равномерно распределенному шуму в интервале $[-1, 1]$.

Матрица смешения $\mathbf{A}$ имеет следующий вид:

$$\mathbf{A} = \begin{bmatrix} 0.56 & 0.79 & -0.37 \\ -0.75 & 0.65 & 0.86 \\ 0.17 & 0.32 & -0.48 \end{bmatrix}.$$

Графики входных сигналов показаны на рис. 10.13, *слева*.

Для разделения воспользуемся пакетной версией правила коррекции (10.104) (см. задачу 10.14). Пакетная обработка была выбрана исходя из соображений ускорения сходимости. В примененном алгоритме использовались следующие состояния.

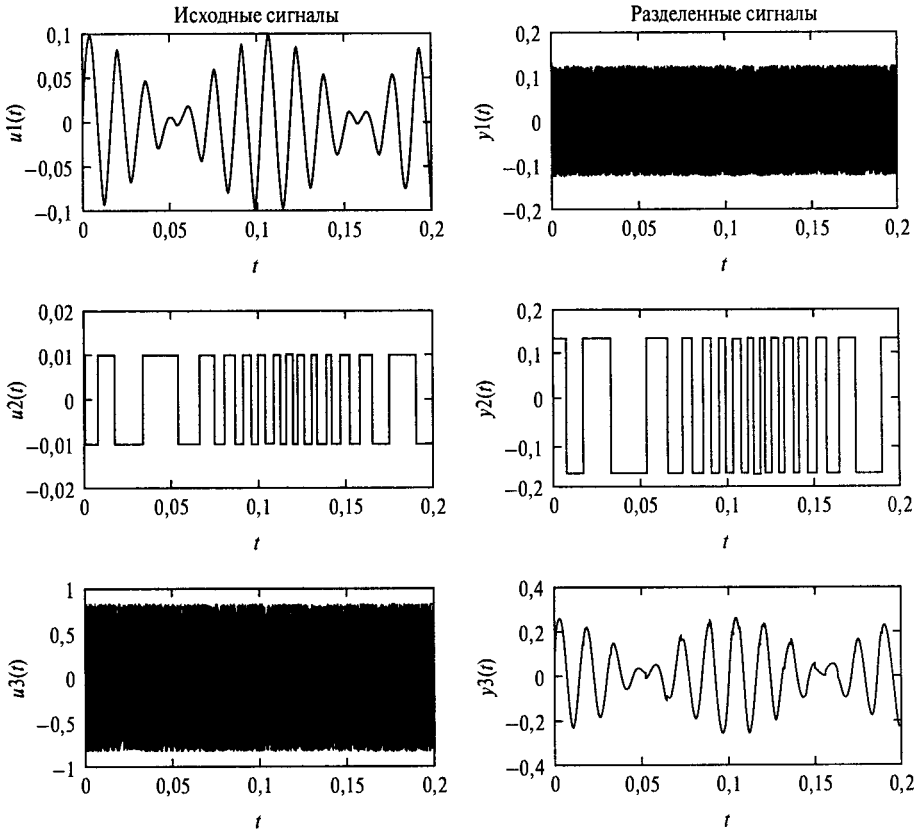


Рис. 10.13. Графики входных сигналов (слева); графики разделенных сигналов (справа)

- *Инициализация.* Для инициализации алгоритма веса матрицы разделения **W** формировались генератором случайных чисел, равномерно распределенных в диапазоне [0, 0.05].
- *Интенсивность обучения.* Параметр скорости обучения был фиксированным и имел значение $\eta = 0{,}1$.
- *Длительность сигнала.* На выходе смесителя сигнал снимался с дискретизацией в 10^{-4} с, при этом множество обучения содержало $N = 65000$ примеров.

На рис. 10.13, *справа* показаны графики сигналов на выходе разделителя после 300 итераций. За исключением некоторого масштабирования и перемешивания порядка сигналов, не существует сколько-нибудь существенных различий между двумя множествами графиков, показанных на рис. 10.13, *слева* и *справа*. Для представленных здесь результатов на этапе инициализации алгоритма использовалась следующая матрица весов **W**:

$$\mathbf{W}(0) = \begin{bmatrix} 0.0109 & 0.0340 & 0.0260 \\ 0.0024 & 0.0467 & 0.0415 \\ 0.0339 & 0.0192 & 0.0017 \end{bmatrix}.$$

Алгоритм сошелся к следующей матрице весов:

$$\mathbf{W} = \begin{bmatrix} 0.2222 & 0.0294 & -0.6213 \\ -10.1932 & -9.8141 & -9.7259 \\ 4.1191 & -1.7879 & -6.3765 \end{bmatrix}.$$

Соответствующими значениями матричного произведения **WA** являются

$$\mathbf{WA} = \begin{bmatrix} -0.0032 & -0.0041 & 0.2413 \\ -0.0010 & -17.5441 & -0.0002 \\ 2.5636 & 0.0515 & -0.0009 \end{bmatrix}.$$

Переставляя строки в матричном произведении так, чтобы порядок выходных сигналов совпадал с порядком входных, можно записать:

$$\mathbf{WA} = \begin{bmatrix} 2.5636 & 0.0515 & -0.0009 \\ -0.0010 & -17.5441 & -0.0002 \\ -0.0032 & -0.0041 & 0.2413 \end{bmatrix}.$$

Первая, вторая и третья строки матричного произведения соответствуют сигналу с амплитудной модуляцией, сигналу с частотной модуляцией и шуму. Диагональные элементы матричного произведения **WA** определяют коэффициенты масштабирования графиков выходных сигналов (см. рис. 10.13, *справа*) по отношению к графикам исходных сигналов (см. рис. 10.13, *слева*).

Для количественной оценки эффективности разделителя можно использовать *глобальный индекс отклонения* (global rejection index), определенный в [37]:

$$J = \sum_{i=1}^{m} \left(\sum_{j=1}^{m} \frac{|p_{ij}|}{\max\limits_{k} |p_{ik}|} - 1 \right) + \sum_{j=1}^{m} \left(\sum_{i=1}^{m} \frac{|p_{ij}|}{\max\limits_{k} |p_{ki}|} - 1 \right),$$

где $\mathbf{P} = \{p_{ij}\} = \mathbf{WA}$. Индекс эффективности J является мерой *диагональности* (diagonality) матрицы **P**. Если матрица **P** является идеально диагональной, то $J = 0$. В матрице **P**, элементы которой не сконцентрированы вдоль главной диагонали, индекс эффективности J будет высоким.

Для графиков, показанных на рис. 10.13, индекс эффективности $J = 0,0606$.

10.13. Оценка максимального правдоподобия

Метод анализа независимых компонентов (т.е. третий вариант принципа Infomax), описанный в предыдущем разделе, является одним из множества методов, которые предлагаются в литературе, для слепого разделения сигналов. Однако в информационно-теоретическом контексте существуют два других метода для решения этой задачи без учителя: методы максимального правдоподобия и максимальной энтропии. В этом разделе мы поговорим о первом из них.

Метод максимального правдоподобия является хорошо зарекомендовавшей себя процедурой статистической оценки, имеющей ряд привлекательных свойств (см. примечание 5 в главе 7). В этой процедуре мы сначала формулируем функцию логарифмического подобия, а затем оптимизируем ее по отношению к вектору параметров рассматриваемой вероятностной модели. В главе 7 уже говорилось о том, что функция подобия является функцией плотности вероятности множества данных в предложенной модели, но рассматривается как функция неизвестных параметров модели. Возвращаясь к рис. 10.9, положим, что $f_U(\cdot)$ — функция плотности вероятности случайного входного вектора $\mathbf{U}$. Тогда функция плотности вероятности вектора наблюдений $\mathbf{X} = \mathbf{AU}$ выхода смесителя будет равна [813]:

$$f_{\mathbf{X}}(\mathbf{x}, \mathbf{A}) = |\det(\mathbf{A})|^{-1} f_{\mathbf{U}}(\mathbf{A}^{-1}\mathbf{x}), \tag{10.112}$$

где $\det(\mathbf{A})$ — определитель матрицы смешения $\mathbf{A}$. Пусть $\mathbf{T} = \{\mathbf{x}_k\}_{k=1}^{N}$ — множество из N независимых реализаций случайного вектора $\mathbf{X}$. Тогда можно записать:

$$f_{\mathbf{X}}(\mathbf{T}, \mathbf{A}) = \prod_{k=1}^{N} f_{\mathbf{X}}(\mathbf{x}_k, \mathbf{A}). \tag{10.113}$$

Считается, что для работы удобнее иметь дело с *нормализованной* (т.е. разделенной на N) версией функции логарифмического подобия:

$$\begin{aligned} \frac{1}{N} \log f_{\mathbf{X}}(\mathbf{T}, \mathbf{A}) &= \frac{1}{N} \sum_{k=1}^{N} \log f_{\mathbf{X}}(\mathbf{x}_k, \mathbf{A}) = \\ &= \frac{1}{N} \sum_{k=1}^{N} \log f_{\mathbf{U}}(\mathbf{A}^{-1}\mathbf{x}_k) - \log |\det(\mathbf{A})|. \end{aligned} \tag{10.114}$$

Пусть $\mathbf{y}=\mathbf{A}^{-1}\mathbf{x}$, а $f_{\mathbf{Y}}(\mathbf{y},\mathbf{W})$ — функция плотности вероятности $\mathbf{Y}$, параметризованная по $\mathbf{W}$. Тогда, признавая, что сумма (10.114) является примером среднего по множеству значения $\log f_{\mathbf{U}}(\mathbf{y}_k)$, можно сказать, что согласно закону больших чисел, с вероятностью 1, при достижении количеством примеров N бесконечности

$$\begin{aligned} L(\mathbf{W}) &= \lim_{N\to\infty}\frac{1}{N}\sum_{k=1}^{N}\log f_{\mathbf{U}}(\mathbf{y}_k)+\log|\det(\mathbf{W})| = \\ &= E[\log f_{\mathbf{U}}(\mathbf{y}_k)]+\log|\det(\mathbf{W})| = \\ &= \int_{-\infty}^{\infty} f_{\mathbf{Y}}(\mathbf{y},\mathbf{W})\log f_{\mathbf{U}}(\mathbf{y})d\mathbf{y}+\log|\det(\mathbf{W})|, \end{aligned} \tag{10.115}$$

где ожидание во второй строке вычисляется по отношению к $\mathbf{Y}$. Величина $L(\mathbf{W})$ является искомой функцией логарифмического подобия. Расписывая

$$f_{\mathbf{U}}(\mathbf{y}) = \left(\frac{f_{\mathbf{U}}(\mathbf{y})}{f_{\mathbf{Y}}(\mathbf{y},\mathbf{W})}\right) f_{\mathbf{Y}}(\mathbf{y},\mathbf{W}),$$

можно переписать функцию $L(\mathbf{W})$ в эквивалентной форме:

$$\begin{aligned} L(\mathbf{W}) &= \int_{-\infty}^{\infty} f_{\mathbf{Y}}(\mathbf{y},\mathbf{W})\log\left(\frac{f_{\mathbf{U}}(\mathbf{y})}{f_{\mathbf{Y}}(\mathbf{y},\mathbf{W})}\right)d\mathbf{y}+ \\ &\quad+\int_{-\infty}^{\infty} f_{\mathbf{Y}}(\mathbf{y},\mathbf{W})\log f_{\mathbf{Y}}(\mathbf{y},\mathbf{W})d\mathbf{y}+\log|\det(\mathbf{W})| = \\ &= -D_{f_{\mathbf{Y}}\|f_{\mathbf{U}}}-h(\mathbf{Y},\mathbf{W})+\log|\det(\mathbf{W})|, \end{aligned} \tag{10.116}$$

где $h(\mathbf{Y},\mathbf{W})$ — дифференциальная энтропия случайного вектора $\mathbf{X}$, параметризованная по $\mathbf{W}$; $D_{f_{\mathbf{Y}}\|f_{\mathbf{U}}}$ — дивергенция Кулбека–Лейблера между $f_Y(\mathbf{y},\mathbf{W})$ и $f_U(\mathbf{y})$. Подставляя (10.76) в (10.116), мы можем упростить выражение для функции логарифмического подобия $L(\mathbf{W})$ [169]:

$$L(\mathbf{W}) = -D_{f_{\mathbf{Y}}\|f_{\mathbf{U}}}-h(\mathbf{X}), \tag{10.117}$$

где $h(\mathbf{X})$ — дифференциальная энтропия случайного вектора $\mathbf{X}$ на входе разделителя. Единственной величиной в (10.117), зависящей от вектора весов $\mathbf{W}$ разделителя, является дивергенция Кулбека–Лейблера $D_{f_{\mathbf{Y}}\|f_{\mathbf{U}}}$. Таким образом, получается, что максимизация функции логарифмического подобия $L(\mathbf{W})$ идентична минимизации дивергенции Кулбека–Лейблера $D_{f_{\mathbf{Y}}\|f_{\mathbf{U}}}$, которая является мерой совпадения выхода разделителя $\mathbf{Y}$ и вектора исходного сигнала $\mathbf{U}$. И это интуитивно понятно.

Связь между максимальным подобием и анализом независимых компонентов

Применяя декомпозицию Пифагора (10.45) к нашей задаче, дивергенцию Кулбека–Леблера для максимального правдоподобия можем представить в следующем виде:

$$D_{f_{\mathbf{Y}}||f_{\mathbf{U}}} = D_{f_{\mathbf{Y}}||\tilde{f}_{\mathbf{Y}}} + D_{\tilde{f}_{\mathbf{Y}}||f_{\mathbf{U}}}. \tag{10.118}$$

Первая дивергенция Кулбека–Лейблера $D_{f_{\mathbf{Y}}||\tilde{f}_{\mathbf{Y}}}$ в правой части (10.118) представляет собой меру *структурного несоответствия* (structural mismatch), которое характеризует метод анализа независимых компонентов. Вторая дивергенция Кулбека–Лейблера является мерой *граничного несоответствия* (marginal mismatch) между граничным распределением выхода разделителя **Y** и распределением вектора исходных сигналов **U**. Таким образом, критерий "глобального" соответствия распределения для метода максимального подобия можно представить следующим образом [23], [169]:

$$\begin{pmatrix}\text{Общее}\\ \text{несоответствие}\end{pmatrix} = \begin{pmatrix}\text{Структурное}\\ \text{несоответствие}\end{pmatrix} + \begin{pmatrix}\text{Граничное}\\ \text{несоответствие}\end{pmatrix}. \tag{10.119}$$

"Структурное несоответствие" относится к структуре распределения, относящейся к множеству независимых переменных; "граничное несоответствие" относится к несоответствию между отдельными граничными распределениями.

При идеальных условиях $\mathbf{W}=\mathbf{A}^{-1}$ (т.е. при совершенно слепом разделении сигнала) как структурное, так и граничное несоответствие исчезает. В этом случае методы максимального правдоподобия и анализа независимых компонентов предлагают одно и то же решение задачи. Идеальная зависимость этих методов показана на рис. 10.14 [23], [172]. На этом рисунке **G** – это множество функций плотности вероятности $f_Y(\mathbf{y})$ случайного вектора **Y** на выходе разделителя; **J** — множество всех независимых распределений вероятности. Как **J**, так и **G** имеет бесконечную размерность. Множество $\mathbf{D} = \{f_{\mathbf{Y}}(\mathbf{y}, \mathbf{W})\}$ является конечным множеством распределений вероятности, измеренных на выходе разделителя. Размерностью множества **D** является m^2, где m — размерность вектора **Y**, а координатная система образована матрицей весов **W**. На рис. 10.14 мы отчетливо видим, что $D_{f_{\mathbf{Y}}||\tilde{f}_{\mathbf{Y}}}$ и $D_{\tilde{f}_{\mathbf{Y}}||f_{\mathbf{U}}}$ имеют минимумы при $\mathbf{W}=\mathbf{A}^{-1}$. Будет интересным доказать, что множества **D** и **J** являются ортогональными в своей точке пересечения, определенной истинной функцией плотности вероятности $f_{\mathbf{U}}(\mathbf{y})$.

Алгоритм слепого разделения источников, основанный на максимальном правдоподобии, должен содержать инструментарий для оценки распределений источников, когда они неизвестны (что, как правило, и происходит). Параметры этой оценки могут адаптироваться по мере адаптации матрицы разделения **W**. Другими словами, необходимо обеспечить *совместную оценку* (joint estimation) матрицы смешения и (некоторых характеристик) распределений входных сигналов [169], [171].

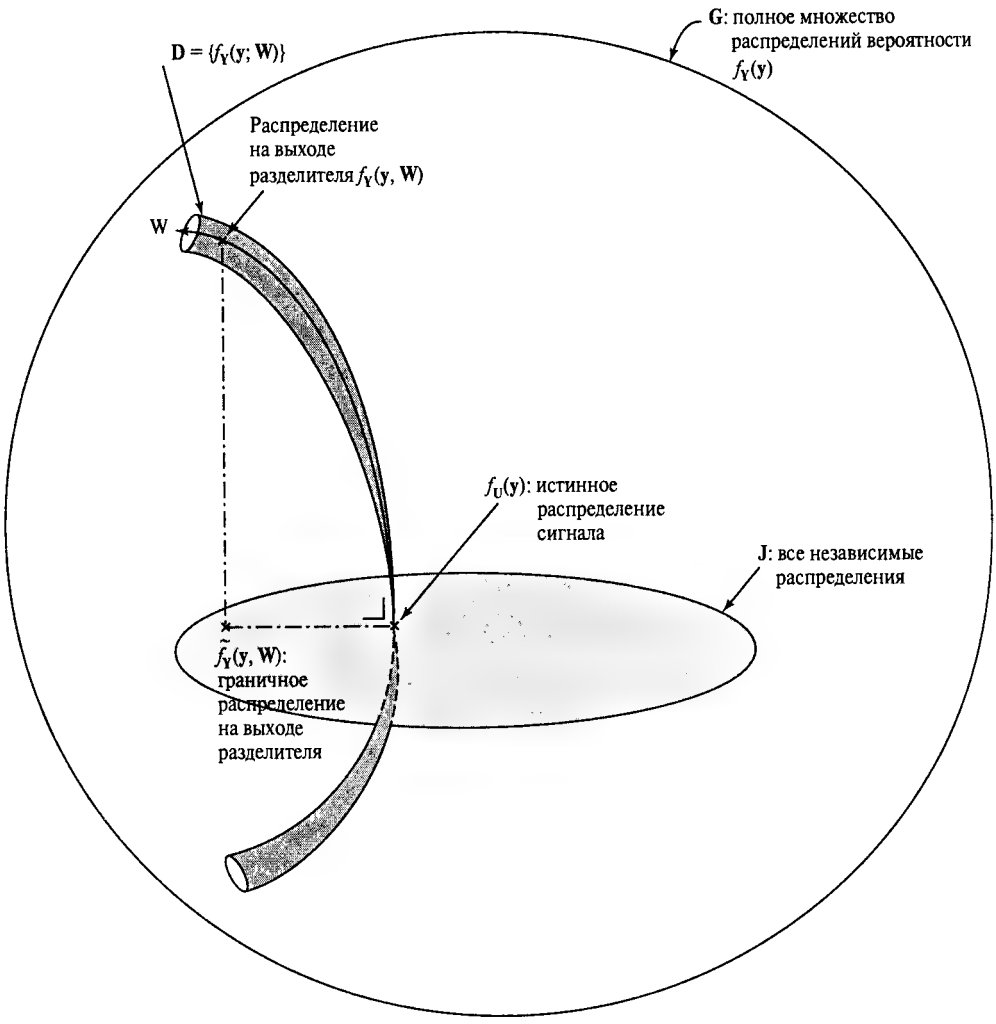


Рис. 10.14. Взаимосвязь между максимальным правдоподобием и анализом независимых компонентов в задаче слепого разделения источников. Метод максимального правдоподобия минимизирует $D_{f_Y||f_U}$, тогда как анализ независимых компонентов минимизирует $D_{f_Y||\tilde{f}_Y}$

Элегантный и хорошо продуманный подход к решению задачи совместной оценки был предложен в [834], [835].

10.14. Метод максимальной энтропии

Метод максимальной энтропии (maximum entropy method) для слепого разделения источников был предложен в [116]. На рис. 10.15 показана блочная диаграмма системы,

основанной на этом методе. Как и раньше, разделитель (demixer) работает с вектором наблюдений **X** для получения выхода **Y=WX**, являющегося оценкой вектора исходных сигналов **U**. Вектор **Y** преобразовывается в вектор **Z** с помощью его прохождения через нелинейное преобразование $\mathbf{G}(\cdot)$, являющееся монотонным и обратимым. Таким образом, в отличие от **Y**, вектор **Z** гарантирует ограниченность дифференциальной энтропии $h(\mathbf{Z})$ для произвольно большого разделителя. Для заданной нелинейности $\mathbf{G}(\cdot)$ метод максимальной энтропии создает оценку исходного входного вектора **U** с помощью максимизации энтропии $h(\mathbf{Z})$ по отношению к **W**. В свете выражения (10.55), выведенного в примере 10.6, видно, что метод максимальной энтропии тесно связан с принципом Infomax[17].

Нелинейность **G** является *диагональным отображением* (diagonal map), описываемым следующим образом:

$$\mathbf{G}: \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{bmatrix} \rightarrow \begin{bmatrix} g_1(y_1) \\ g_2(y_2) \\ \cdots \\ g_m(y_m) \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \cdots \\ z_m \end{bmatrix}. \tag{10.120}$$

Исходя из этого, можно записать:

$$\mathbf{Z} = \mathbf{G}(\mathbf{Y}) = \mathbf{G}(\mathbf{WAU}). \tag{10.121}$$

Так как нелинейность $\mathbf{G}(\cdot)$ является обратимой, входной вектор **U** можно выразить в терминах выходного вектора **Z**:

$$\mathbf{U} = \mathbf{A}^{-1}\mathbf{W}^{-1}\mathbf{G}^{-1}(\mathbf{Z}) = \mathbf{\Psi}(\mathbf{Z}), \tag{10.122}$$

где $\mathbf{G}^{-1}$ — *обратная нелинейность* (inverse nonlinearity):

$$\mathbf{G}^{-1}: \begin{bmatrix} z_1 \\ z_2 \\ \cdots \\ z_m \end{bmatrix} \rightarrow \begin{bmatrix} g_1^{-1}(z_1) \\ g_2^{-1}(z_2) \\ \cdots \\ g_m^{-1}(z_m) \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{bmatrix}. \tag{10.123}$$

---

[17] В [116] авторы назвали свой метод слепого разделения источников словом Infomax в свете выражения (10.55), определяющего взаимосвязь между энтропией $H(\mathbf{Y})$ и взаимной информацией $I(\mathbf{Y};\mathbf{X})$. Тем не менее более предпочтителен термин "метод максимальной энтропии", так как в этом методе в действительности максимизируется энтропия $h(\mathbf{Z})$, где $\mathbf{Z} = \mathbf{G}(\mathbf{Y})$. Не путайте метод максимальной энтропии для задачи слепого разделения источников по Беллу и Седжновскому с методом максимальной энтропии по Бургу [164] для спектрального анализа.

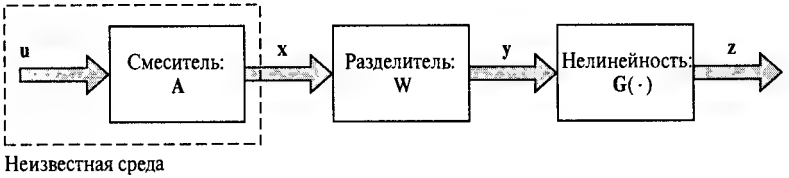


Рис. 10.15. Блочная диаграмма метода максимальной энтропии для задачи слепого разделения источников. Векторы **u**, **x**, **y** и **z** являются значениями случайных векторов **U**, **X**, **Y** и **Z** соответственно

Функция плотности вероятности выходного вектора **Z** определяется в терминах функции плотности вероятности входного вектора **U** следующим образом:

$$f_{\mathbf{Z}}(\mathbf{z}) = \frac{f_{\mathbf{U}}(\mathbf{u})}{|\det(\mathbf{J}(\mathbf{u}))|}\Big|_{\mathbf{u}=\Psi(\mathbf{Z})}, \tag{10.124}$$

где $\det(\mathbf{J}(\mathbf{u}))$ — определитель матрицы Якобиана $\mathbf{J}(\mathbf{u})$, ij-м элементом которой является

$$J_{ij} = \frac{\partial z_i}{\partial u_j}. \tag{10.125}$$

Исходя из этого, энтропию случайного вектора **Z** на выходе нелинейности **G** можно записать в соответствии с формулой

$$\begin{aligned} h(\mathbf{Z}) &= -E[\log f_{\mathbf{Z}}(\mathbf{z})] = -E\left[\log\left(\frac{f_{\mathbf{U}}(\mathbf{u})}{|\det(\mathbf{J}(\mathbf{u}))|}\right)\right]_{\mathbf{u}=\Psi(\mathbf{z})} = \\ &= -D_{f_{\mathbf{U}}||\det \mathbf{J}|}, \quad \text{вычисленной в точке } \mathbf{u} = \Psi(\mathbf{z}). \end{aligned} \tag{10.126}$$

Мы видим, что максимизация энтропии $h(\mathbf{Z})$ эквивалентна минимизации дивергенции Кулбека–Лейблера между $f_{\mathbf{U}}(\mathbf{u})$ и функцией плотности вероятности **U**, задаваемой определителем $|\det(\mathbf{J}(\mathbf{u}))|$.

Теперь предположим, что случайная переменная Z_i (т.е. i-й компонент вектора **Z**) *равномерно распределена* на интервале [0, 1] для всех i. Согласно примеру 10.1, энтропия $h(\mathbf{Z})$ в этом случае равна нулю. Следовательно, из (10.126) можно заключить, что

$$f_{\mathbf{U}}(\mathbf{u}) = |\det(\mathbf{J}(\mathbf{u}))|. \tag{10.127}$$

При идеальных условиях ($\mathbf{W}=\mathbf{A}^{-1}$) это соотношение сводится к

$$f_{U_i}(u_i) = \frac{\partial z_i}{\partial y_i}\bigg|_{z_i = g(u_i)} \quad \text{для всех } i. \tag{10.128}$$

И наоборот, можно утверждать, что если удовлетворяется (10.128), то максимизация $h(\mathbf{Z})$ приводит к тому, что $\mathbf{W} = \mathbf{A}^{-1}$, и, таким образом, достигается слепое разделение источников.

Теперь можно подвести итоги, полученные для метода максимальной энтропии решения задачи слепого разделения источников [116].

Пусть нелинейность на выходе разделителя (см. рис. 10.15) можно определить в терминах исходного распределения источников следующим образом:

$$z_i = g_i(y_i) = \int_{-\infty}^{z_i} f_{U_i}(u_i)du_i, \quad i = 1, 2, \ldots, m. \tag{10.129}$$

Тогда максимизация энтропии случайной переменной $\mathbf{Z}$ *на выходе нелинейной зависимости* $\mathbf{G}$ *эквивалентна достижению соотношения* $\mathbf{W}=\mathbf{A}^{-1}$, *что соответствует полному слепому разделению источников.*

Методы максимальной энтропии и максимального подобия для задачи слепого разделения источников являются эквивалентными, при условии, что случайная переменная Z_i равномерно распределена на интервале [0, 1] для всех i [171]. Для доказательства этого соотношения можно использовать *правило цепочки* (chain rule) и переписать (10.125) в эквивалентной форме:

$$J_{ij} = \sum_{k=1}^{m} \frac{\partial z_i}{\partial y_i} \frac{\partial y_i}{\partial x_k} \frac{\partial x_k}{\partial u_j} = \sum_{k=1}^{m} \frac{\partial z_i}{\partial y_i} w_{ik} a_{kj}. \tag{10.130}$$

Следовательно, матрица Якобиана $\mathbf{J}$ может быть выражена как

$$\mathbf{J} = \mathbf{DWA},$$

где $\mathbf{D}$ — диагональная матрица:

$$\mathbf{D} = \operatorname{diag}\left(\frac{\partial z_1}{\partial y_1}, \frac{\partial z_2}{\partial y_2}, \ldots, \frac{\partial z_m}{\partial y_m}\right).$$

Исходя из этого,

$$|\det(\mathbf{J})| = |\det(\mathbf{WA})| \prod_{i=1}^{m} \frac{\partial z_i}{\partial y_i}. \tag{10.131}$$

Оценка функции плотности вероятности $f_{\mathbf{U}}(\mathbf{u})$, параметризованная по матрице весов $\mathbf{W}$ и нелинейности $\mathbf{G}$, в свете (10.131) может быть формально переписана в следующем виде [906]:

$$f_{\mathbf{U}}(\mathbf{u}|\mathbf{W},\mathbf{G}) = |\det(\mathbf{WA})| \prod_{i=1}^{m} \frac{\partial g_i(y_i)}{\partial y_i}. \tag{10.132}$$

Отсюда видно, что при этом условии максимизация функции логарифмического подобия $\log f_{\mathbf{U}}(\mathbf{u}|\mathbf{W},\mathbf{G})$ эквивалентна максимизации энтропии $h(\mathbf{Z})$ в задаче слепого разделения источников. А это значит, что методы максимальной энтропии и максимального правдоподобия эквивалентны.

Алгоритм обучения для слепого разделения источников

Возвращаясь ко второй строке выражения (10.126), отметим, что при фиксированном распределении источника максимизация энтропии $h(\mathbf{z})$ требует максимизации математического ожидания делителя $\log|\det(\mathbf{J}(\mathbf{u}))|$ по отношению к матрице весов $\mathbf{W}$. Применяя для этой цели адаптивный алгоритм, можно рассмотреть в качестве целевой функции следующую:

$$\Phi = \log|\det(\mathbf{J})|. \tag{10.133}$$

Подставляя (10.131) в (10.133), получим:

$$\Phi = \log|\det(\mathbf{A})| + \log|\det(\mathbf{W})| + \sum_{i=1}^{m} \log\left(\frac{\partial z_i}{\partial y_i}\right). \tag{10.134}$$

Дифференцируя Φ по матрице весов $\mathbf{W}$ разделителя, получим (см. задачу 10.16):

$$\frac{\partial \Phi}{\partial \mathbf{W}} = \mathbf{W}^{-T} + \sum_{i=1}^{m} \frac{\partial}{\partial \mathbf{W}} \log\left(\frac{\partial z_i}{\partial y_i}\right). \tag{10.135}$$

Для того чтобы продолжить работу, требуется определить нелинейность, формирующую выход разделителя. В качестве простой формы такой нелинейности можно использовать логистическую функцию

$$z_i = g(y_i) = \frac{1}{1+e^{-y_i}}, \quad i = 1, 2, \ldots, m. \tag{10.136}$$

На рис. 10.16 представлены графики нелинейности и функции, обратной ей. На этом рисунке видно, что логистическая функция удовлетворяет основным требованиям монотонности и обратимости, налагаемым задачей слепого разделения источников. Подставляя (10.136) в (10.135), получим:

$$\frac{\partial \Phi}{\partial \mathbf{W}} = \mathbf{W}^{-T} + (\mathbf{1} - 2\mathbf{z})\mathbf{x}^T,$$

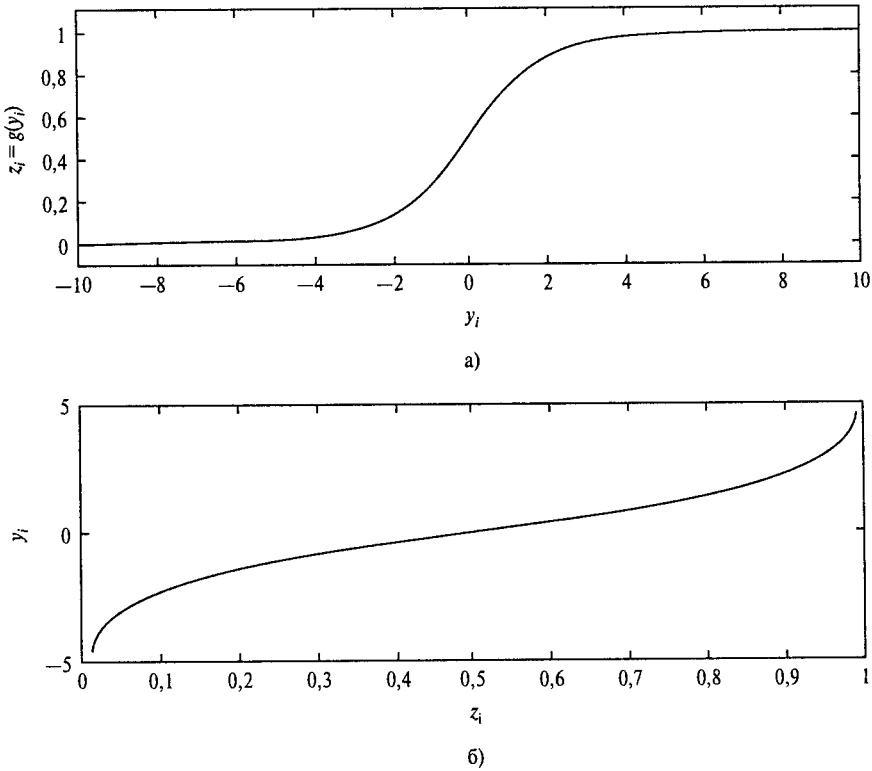


Рис. 10.16. Логистическая (сигмоидальная) функция $z_i = g(y_i) = \frac{1}{1+e^{-y_i}}$ (а); функция, обратная к сигмоидальной $y_i = g^{-1}(z_i)$ (б)

где $\mathbf{x}$ — полученный вектор сигнала; z — нелинейно-преобразованный вектор выхода; $\mathbf{1}$ — вектор, составленный из единиц.

Целью алгоритма обучения является максимизация энтропии $h(\mathbf{Z})$. Следовательно, задействуя метод наискорейшего спуска, получим формулу для корректировки матрицы весов $\mathbf{W}$ [116]:

$$\Delta\mathbf{W} = \eta\frac{\partial\Phi}{\partial\mathbf{W}} = \eta(\mathbf{W}^{-T} + (\mathbf{1} - 2\mathbf{z})\mathbf{x}^{\mathbf{T}}), \tag{10.137}$$

где η — параметр скорости обучения. В анализе независимых компонентов можно избежать необходимости обращения транспонированной матрицы весов $\mathbf{W}^T$, используя натуральный градиент. Это эквивалентно перемножению выражения (10.137) на матричное произведение $\mathbf{W}^T\mathbf{W}$. Это оптимальное масштабирование приводит к искомой формуле коррекции весов:

$$\Delta \mathbf{W} = \eta(\mathbf{W}^{-T} + (\mathbf{1} - 2\mathbf{z})\mathbf{x}^T)\mathbf{W}^T\mathbf{W} = \eta(\mathbf{I} + (\mathbf{1} - 2\mathbf{z})(\mathbf{W}\mathbf{x})^T)\mathbf{W} = \\ = \eta(\mathbf{I} + (\mathbf{1} - 2\mathbf{z})\mathbf{y}^T)\mathbf{W}, \tag{10.138}$$

где вектор $\mathbf{y}$ — выход разделителя. Алгоритм обучения для вычисления матрицы весов $\mathbf{W}$ имеет следующий вид:

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \eta(\mathbf{I} + (\mathbf{1} - 2\mathbf{z}(n))\mathbf{y}^T(n))\mathbf{W}(n). \tag{10.139}$$

Этот алгоритм инициализируется значением $\mathbf{W}(0)$, выбираемым из равномерно распределенного множества малых чисел.

Теоретические и экспериментальные исследования показали, что применение алгоритма обучения (10.139) ограничено разделением источников, имеющих субгауссово распределение[18] [116]. Это ограничение является прямым следствием использования в качестве нелинейности логистической функции (см. рис. 10.15). В частности, логистическая функция предполагает наличие априорных знаний об источнике сигнала, а именно его супергауссовой формы. Однако в методе максимальной энтропии не существует ничего, что ограничивало бы его использованием только логистической функции. То же касается и метода максимального правдоподобия, рассмотренного ранее. Применение метода максимальной энтропии может быть расширено на более широкий спектр распределений входного сигнала за счет изменения алгоритма обучения (10.138) так, чтобы он проводил совместную оценку распределения источников и матрицы смешения. Аналогичное требование выдвигалось и в отношении метода максимального правдоподобия в предыдущем разделе.

10.15. Резюме и обсуждение

В этой главе было введено понятие взаимной информации, уходящее корнями в теорию информации Шеннона. Это понятие послужило основой для статистических механизмов самоорганизации. Взаимная информация между входным и выходным процессами имеет ряд уникальных свойств, которые позволяют применить ее в ка-

---

[18] Случайная переменная X называется субгауссовой [122], если выполняются следующие условия:
она равномерно распределена;
ее функция плотности вероятности $f_X(x)$ может быть представлена в форме $\exp(-g(x))$, где $g(x)$ — гладкая функция, дифференцируемая всюду (возможно, за исключением начала координат), а $g(x)$ и $g'(x)/x$ строго возрастают на интервале $0 < x < \infty$.
Примером такой функции может стать $g(x) = |x|^\beta$, где $\beta > 2$.
Если же на интервале $0 < x < \infty$ функция $g'(x)/x$ строго убывает, но сохраняются все остальные описанные свойства, такая случайная переменная называется супергауссовой [122]. Примером такой функции может быть $g(x) = |x|^\beta$, $\beta < 2$.
Иногда в качестве индикатора суб- или супергауссового распределения используется знак эксцесса

$$K_4(x) = \frac{E[X^4]}{(E[X^2])^2} - 3.$$

Если эксцесс отрицателен, случайная переменная называется субгауссовой, а если положителен — супергауссовой.

честве оптимизируемой целевой функции в самоорганизующемся обучении. Из обсуждения, приведенного в этой главе, можно вывести некоторые важные принципы самоорганизации.

- *Принцип максимума взаимной информации* (Infomax) [653], [654]. Этот принцип в своей основной форме хорошо подходит для создания самоорганизующихся моделей и карт признаков.
- *Первый вариант принципа Infomax* [114] хорошо соответствует обработке изображений, целью которой является обнаружение свойств зашумленного входного сенсорного сигнала, связного по времени и пространству.
- *Второй вариант принципа Infomax* [1068] нашел свое применение в обработке двух изображений, целью которой является максимизация пространственных различий между соответствующими областями двух различных изображений (видов) интересующего объекта.
- *Третий вариант принципа Infomax*, или *анализ независимых компонентов* [205], уходит своими корнями в [90], [91]. Однако только в [205] была впервые изложена строгая формулировка анализа независимых компонентов.
- *Метод максимальной энтропии* [116] также связан с принципом Infomax. Максимальная энтропия является эквивалентом максимального подобия [171].

Анализ независимых компонентов и метод максимальной энтропии являются двумя альтернативными методами слепого разделения сигнала, каждый из которых имеет свои собственные атрибуты. Алгоритм слепого разделения сигнала, базирующийся на методе максимальной энтропии, прост в реализации, в то время как соответствующий алгоритм, основанный на анализе независимых компонентов, более сложен для формулировки, однако имеет более широкую область применения.

Примером из нейробиологии, который связывают с задачей слепого разделения сигнала, является явление, получившее название "эффекта вечеринки". Это явление связано с выдающейся способностью человека избирательно настраиваться на интересующий звуковой сигнал в зашумленной среде и отслеживать его. Как уже говорилось в главе 2, основополагающая нейробиологическая модель, привлекаемая к решению этой сложной задачи обработки сигнала, является гораздо более сложной, чем идеализированная модель, показанная на рис. 10.9. Нейробиологическая модель содержит как временные, так и пространственные формы обработки, которые требуются для отделения неизвестной задержки, реверберации и шума. Теперь, когда мы уяснили основные вопросы, касающиеся нейронных решений задачи слепого разделения сигнала, пожалуй, стоит перейти к реальным задачам, в которых возникают эффекты, подобные "эффекту вечеринки".

Еще одной открытой областью исследований, которой уделяется пристальное внимание, является задача *слепого восстановления сигнала* или обращенной свертки (blind deconvolution). Слепое восстановление сигнала представляет собой операцию

обработки сигнала, обратную свертке, состоящей в линейной, инвариантной ко времени системе обработки входного сигнала. Если говорить более конкретно, то в обычной задаче восстановления сигнала известны как выходной сигнал, так и сама система, а требуется восстановить исходный сигнал. При слепом восстановлении (т.е. обращении свертки без учителя) известен *только* выходной сигнал и иногда информация о статистике входного сигнала, а требуется найти входной сигнал и (или) систему. Совершенно понятно, что задача слепого восстановления сигнала является более сложной, чем задача обычного обращения свертки. Несмотря на то что слепому восстановлению сигнала уделяется больше внимания в литературе [436], наше понимание информационно-теоретического подхода к слепому восстановлению сигнала, аналогичное задаче слепого разделения сигнала, находится на ранней стадии развития [264]. Более того, задача *слепого уравнивания* (blind equalization) канала, такого как канал мобильной связи, требует своего эффективного решения не меньше, чем задача слепого разделения источников.

Подводя итог сказанному, можно отметить, что *слепой адаптации* (blind adaptation), будь то в контексте разделения источников или развертки, предстоит пройти еще долгий путь до того момента, когда она достигнет уровня развития, сравнимого с задачами обучения с учителем.

Задачи

Принцип максимума энтропии

10.1. Несущее множество (support) случайной переменной X (т.е. диапазон значений, для которых она ненулевая) определяется отрезком $[a, b]$. Других ограничений на эту случайную переменную не налагается. Какова максимальная энтропия распределения этой случайной переменной? Обоснуйте свой ответ.

Взаимная информация

10.2. Выведите свойства взаимной информации $I(X;Y)$ между двумя случайными величинами X и Y с непрерывным диапазоном значений (см. раздел 10.4).

10.3. Рассмотрим случайный входной вектор $\mathbf{X}$, составленный из первичных компонентов $\mathbf{X}_1$ и контекстных компонентов $\mathbf{X}_2$. Определим:

$$Y_i = \mathbf{a}_i^T \mathbf{X}_1,$$
$$Z_i = \mathbf{b}_i^T \mathbf{X}_2.$$

Как взаимная информация между $\mathbf{X}_1$ и $\mathbf{X}_2$ связана со взаимной информацией между Y_i и Z_i? Предположим, что модель вероятности $\mathbf{X}$ определяется многомерным гауссовым распределением:

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{m/2}(\det \boldsymbol{\Sigma})^{1/2}} \exp((\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})),$$

где $\boldsymbol{\mu}$ — средний вектор $\mathbf{X}$; $\boldsymbol{\Sigma}$ — матрица ковариации.

10.4. В этой задаче исследуем использование относительной энтропии, или дивергенции Кулбека–Лейблера, с целью вывода алгоритма обучения с учителем для многослойного персептрона [108], [478]. Для примера рассмотрим многослойный персептрон, состоящий из входного, скрытого и выходного слоя. При подаче на вход системы данного примера α выходу нейрона k выходного слоя соответствует следующая вероятностная интерпретация:

$$y_{k|\alpha} = p_{k|\alpha}.$$

Пусть $q_{k|\alpha}$ — истинное значение условной вероятности того, что для данного примера α предположение k истинно. Тогда относительная энтропия многослойного персептрона определяется следующей формулой:

$$D_{p||q} = \sum_{\alpha} p_{\alpha} \sum_{k} \left(q_{k|\alpha} \log \left(\frac{q_{k|\alpha}}{p_{k|\alpha}} \right) + (1 - q_{k|\alpha}) \log \left(\frac{1 - q_{k|\alpha}}{1 - p_{k|\alpha}} \right) \right),$$

где p_{α} — априорная вероятность возникновения события α.

Используя $D_{p||q}$ в качестве оптимизируемой функции стоимости, выведите правило обучения для многослойного персептрона.

Принцип Infomax

10.5. Рассмотрим два канала, выходы которых представлены случайными переменными X и Y. Требуется максимизировать взаимную информацию между X и Y. Покажите, что это требование удовлетворяется, если выполняются два условия.

(а) Вероятности возникновения событий X и Y равны $0,5$.

(б) Распределение совместной вероятности X и Y сконцентрировано в небольшой области пространства вероятности.

10.6. Рассмотрим зашумленную модель на рис. 10.17 с m узлами источника во входном слое сети, состоящим из двух нейронов. Оба нейрона являются линейными. Входные сигналы обозначены символами $X_1, X_2, \ldots, X_m$, а выходные — Y_1 и Y_2. Можно сделать следующие допущения.

- Компоненты аддитивного шума N_1 и N_2 на выходе сети имеют гауссово распределение с нулевым средним и дисперсией σ_N^2. Они не коррелированы друг с другом.
- Каждый из источников шума не коррелирован со входными сигналами.

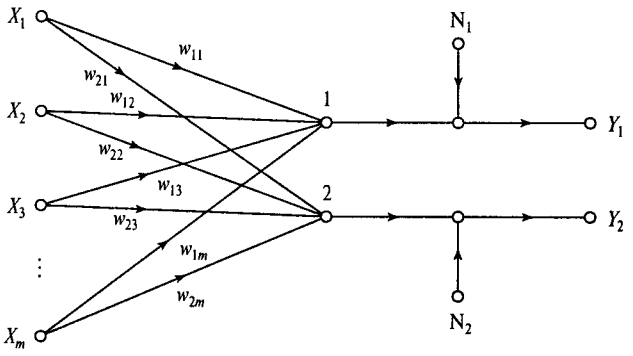


Рис. 10.17.

- Выходные сигналы Y_1 и Y_2 являются гауссовыми случайными переменными с нулевым средним.

(а) Найдите взаимную информацию $I(\mathbf{Y}; \mathbf{X})$ между выходным $\mathbf{Y}=[Y_1, Y_2]^T$ и входным вектором $\mathbf{X}=[X_1, X_2, \ldots, X_m]^T$.

(б) Используя результаты части (а), исследуйте баланс между избыточностью и разнообразием при следующих условиях [653].

1. Дисперсия шума σ_N^2 значительно превосходит дисперсии Y_1 и Y_2.
2. Дисперсия шума σ_N^2 значительно меньше дисперсии Y_1 и Y_2.

10.7. В варианте принципа Infomax [114], описанном в разделе 10.9, целью является максимизация взаимной информации $I(Y_a; Y_b)$ между выходами Y_a и Y_b зашумленной нейронной системы, являющимися реакцией системы на входные векторы $\mathbf{X}_a$ и $\mathbf{Y}_b$. В другом подходе [114] ставится другая цель: максимизировать взаимную информацию $I\left(\frac{Y_a+Y_b}{2}; S\right)$ между средним выходов Y_a и Y_b и компонентом S рассматриваемого сигнала, общим для этих двух выходов.

Используя модель, описываемую выражениями (10.59) и (10.60), выполните следующее.

(а) Покажите, что

$$I\left(\frac{Y_a + Y_b}{2}; S\right) = \log\left(\frac{\text{var}[Y_a + Y_b]}{\text{var}[N_a + N_b]}\right),$$

где N_a и N_b — компоненты шума в Y_a и Y_b соответственно.

(б) Продемонстрируйте интерпретацию этой взаимной информации в качестве отношения сигнал–плюс–шум/шум.

Анализ независимых компонентов

10.8. Проведите детальное сравнение анализа главных компонентов (см. главу 8) с анализом независимых компонентов.

10.9. Анализ независимых компонентов может использоваться как шаг предварительной обработки в *приближенном анализе данных* (approximate data analysis) перед *детектированием* (detection) и классификацией [205]. Обсудите то свойство анализа независимых компонентов, которое может использоваться в этом приложении.

10.10. *Теорема Дармуса* (Darmois) гласит, что сумма независимых переменных может иметь гауссово распределение только в том случае, если все они сами являются гауссовыми [239]. С помощью анализа независимых компонентов докажите эту теорему.

10.11. На практике алгоритмическая интерпретация анализа независимых компонентов может достигать "*максимально большой статистической независимости*" (as statistically independent as possible). Сопоставьте решение задачи слепого разделения сигналов на основе этого алгоритма с решением, полученным с использованием метода декорреляции. Предполагается, что матрица ковариации вектора наблюдений является несингулярной.

10.12. Ссылаясь на схему, приведенную на рис. 10.9, покажите, что минимизация взаимной информации между любыми двумя компонентами выхода *разделителя* (demixer) $\mathbf{Y}$ эквивалентна минимизации дивергенции Кулбека–Лейблера между параметризованной функцией плотности вероятности $f_Y(\mathbf{y},\mathbf{W})$ и соответствующим факториальным распределением $\tilde{f}_{\mathbf{Y}}(\mathbf{y}, \mathbf{W})$.

10.13. Адаптивный алгоритм слепого разделения сигналов (10.104) обладает двумя важными свойствами: свойством эквивариантности и тем свойством, что формируемая матрица весов $\mathbf{W}$ является несингулярной. Первое свойство достаточно подробно рассматривалось в разделе 10.11. В этой задаче мы займемся вторым свойством.

Предполагая, что начальное значение $\mathbf{W}(0)$, используемое для инициализации алгоритма (10.104), удовлетворяет условию $|\det(\mathbf{W}(0))| \neq 0$, покажите, что

$$|\det(\mathbf{W}(0))| \neq 0 \text{ для всех } n.$$

Это необходимое и достаточное условие для того, чтобы матрица $\mathbf{W}(n)$ была несингулярной для всех n.

10.14. В этой задаче мы сформулируем пакетную версию алгоритма слепого разделения сигналов (10.104). В частности, можно записать:

$$\Delta\mathbf{W} = \eta\left(\mathbf{I} - \frac{1}{N}\boldsymbol{\Phi}(\mathbf{Y})\mathbf{Y}^T\right)\mathbf{W},$$

где

$$\mathbf{Y} = \begin{bmatrix} y_1(1) & y_1(2) & \dots & y_1(N) \\ y_2(1) & y_2(2) & \dots & y_2(N) \\ \dots & \dots & \dots & \dots \\ y_m(1) & y_m(2) & \dots & y_m(N) \end{bmatrix}$$

и

$$\Phi(\mathbf{Y}) = \begin{bmatrix} \varphi(y_1(1)) & \varphi(y_1(2)) & \dots & \varphi(y_1(N)) \\ \varphi(y_2(1)) & \varphi(y_2(2)) & \dots & \varphi(y_2(N)) \\ \dots & \dots & \dots\dots & \\ \varphi(y_m(1)) & \varphi(y_m(2)) & \dots & \varphi(y_m(N)) \end{bmatrix},$$

где N — количество доступных точек данных. Обоснуйте описанную формулу коррекции $\Delta\mathbf{W}$ матрицы весов $\mathbf{W}$.

Метод максимальной энтропии

10.15. Рассмотрим рис. 10.15, на котором

$$\mathbf{Y} = \mathbf{WX},$$

где

$$\mathbf{Y} = [Y_1, Y_2, \dots, Y_m]^T,$$
$$\mathbf{X} = [X_1, X_2, \dots, X_m]^T,$$

а $\mathbf{W}$ — матрица весов размерности $m \times m$. Пусть

$$\mathbf{Z} = [Z_1, Z_2, \dots, Z_m]^T,$$

где

$$Z_k = \varphi(Y_k), k = 1, 2, \dots, m.$$

(а) Покажите, что совместная энтропия $\mathbf{Z}$ связана с дивергенцией Кулбека-Лейблера $D_{f||\tilde{f}}$ следующим соотношением:

$$h(\mathbf{Z}) = -D_{f||\tilde{f}} - D_{\tilde{f}||q},$$

где $D_{\tilde{f}||q}$ — дивергенция Кулбека-Лейблера между (а) функцией плотности вероятности статистически независимой (т.е. факторизированной) версии выходного вектора $\mathbf{Y}$ и (б) "функцией плотности вероятности", являющейся результатом выражения $\prod_{i=1}^{m} q(y_i)$.

(б) Как изменится формула для $h(\mathbf{Z})$, если $q(y_i)$ будет равна функции плотности вероятности истинного выхода U_i для всех i?

10.16. (а) Из выражения (10.134) выведите выражение (10.135).

(б) Для логистической функции (10.136) покажите, что с помощью (10.135) можно получить формулу (10.137).

ГЛАВА 11

Стохастические машины и их аппроксимации в статистической механике

11.1. Введение

Последний класс самоорганизующихся обучаемых систем, который рассматривается в данной книге, черпает свои идеи в статистической механике. *Статистическая механика* (statistical mechanics) занимается формальным изучением макроскопических свойств равновесия в крупных системах, состоящих из элементов, подверженных микроскопическим законам механики. Главной целью статистической механики является управление термодинамическими свойствами макроскопических объектов, начиная с движения микроскопических частиц, таких как атомы и электроны [611], [814]. Количество степеней свободы, с которыми приходится иметь дело в таких системах, невероятно велико. Это и является причиной применения для решения задач именно вероятностных методов. Как и в теории информации Шеннона, главную роль в статистической механике играет концепция энтропии. Чем более упорядоченной является система или чем более концентрировано распределение вероятности, тем меньшей является энтропия. Аналогично, можно заявить, что чем сильнее беспорядок в системе или чем ближе распределение вероятности к равномерному, тем выше энтропия. В [512] было показано, что энтропию можно использовать не только в качестве отправной точки при формулировке статистических выводов, но и для генерирования распределения Гиббса, лежащего в основе статистической механики.

Интерес к использованию статистической механики при изучении нейронных сетей был проявлен еще в ранних работах [223], [229]. *Машина Больцмана* (Boltzmann machine) [9], [464], [465] была, пожалуй, первой многослойной обучаемой машиной, вдохновленной идеями статистической механики. Свое название эта машина получила в признание формальной эквивалентности между основополагающими работами Больцмана в статистической термодинамике и динамическим поведением сетей. В своей основе машина Больцмана представляет собой устройство модели-

рования исследуемого распределения вероятности на основе множества данных, из чего можно вывести условное распределение, используемое в таких задачах, как распознавание образов и классификация множеств. К сожалению, процесс обучения в машине Больцмана протекает чрезвычайно медленно. Это стало причиной того, что машина Больцмана подверглась модификациям, результатом которых стало появление новых стохастических машин. Большая часть материала настоящей главы будет посвящена их описанию.

Структура главы

Главу формально можно разбить на три части. Первая часть состоит из разделов 11.2–11.6. В разделе 11.2 представлен краткий обзор задач статистической механики. В разделе 11.3 предложен обзор особого типа стохастических процессов, получивший название цепей Маркова (Markov chain). Его часто можно встретить при изучении статистической механики. В разделах 11.4–11.6 последовательно рассматриваются три метода стохастического моделирования: алгоритм Метрополиса (Metropolis), метод моделирования отжига (simulated annealing) и выборки Гиббса (Gibbs sampling).

Вторая часть главы, состоящая из разделов 11.7–11.9, описывает три типа стохастических машин: машина Больцмана, сигмоидальные сети доверия (sigmoid belief network) и стохастическая машина Гельмгольца (Helmholtz).

В последней части главы (разделы 11.10–11.13) основное внимание уделяется вопросам аппроксимации стохастических машин. Эта аппроксимация основана на идеях *теории среднего поля* (mean-field theory) в статистической механике. В разделе 11.10 сформулированы основные положения этой теории. В разделе 11.11 рассматривается простейшая теория среднего поля машины Больцмана, за чем в разделе 11.12 последует более принципиальный подход к теории среднего поля сигмоидальных сетей доверия. В разделе 11.13 описывается детерминированный отжиг как аппроксимация моделирования отжига.

Как всегда, глава завершается общими выводами и рассуждениями.

11.2. Статистическая механика

Рассмотрим физическую систему с множеством степеней свободы, которая может находиться в одном из большого количества возможных состояний. Обозначим символом p_i вероятность нахождения системы в определенном состоянии i, где

$$p_i \geq 0 \text{ для всех } i \tag{11.1}$$

и

$$\sum_i p_i = 1. \tag{11.2}$$

Пусть E_i — *энергия* системы, когда она находится в состоянии i. Фундаментальный результат статистической механики утверждает, что если система находится в термальном равновесии с окружающей средой, состояние i возникает с вероятностью

$$p_i = \frac{1}{Z} \exp\left(-\frac{E_i}{k_B T}\right), \tag{11.3}$$

где T — *абсолютная температура* в градусах Кельвина; k_B — *константа Больцмана*; Z — константа, не зависящая от конкретного состояния. Нулевой градус Кельвина соответствует температуре $-273°$ по Цельсию. Константа Больцмана равна $k_B = 1{,}38 \times 10^{-23}$ Джоуля/Кельвин.

Уравнение (11.2) определяет условие нормализации вероятностей. Подставляя это условие в (11.3), получим:

$$Z = \sum_i \exp\left(-\frac{E_i}{k_B T}\right). \tag{11.4}$$

Нормализующая величина Z называется *суммой по всем состояниям* (sum over states) или *функцией разбиения* (partition function). (Для обозначения этой величины принято использовать именно символ Z, так как по-немецки она называется *Zustadsumme*.) Распределение вероятности (11.3) называется *каноническим распределением* (canonical distribution) или *распределением Гиббса* (Gibbs distribution)[1]. Экспоненциальный множитель $\exp(-E_i/k_B T)$ называют *коэффициентом Больцмана* (Boltzmann factor).

При работе с распределением Гиббса очень важно учесть следующие вопросы.

1. Состояния с низкой энергией более вероятны, чем состояния с высокой энергией.
2. При понижении температуры T вероятность концентрируется на небольшом подмножестве состояний с низкой энергией.

---

[1] Термин "каноническое распределение" был введен в работе Гиббса [348]. В первой части своей книги *Элементарные принципы статистической механики* он написал:

"Распределение, представленное формулой $P = \exp\left(\frac{\psi - E}{H}\right)$, где H и ψ — константы, при этом H — положительная, представляет собой самый простой возможный случай, так как оно имеет следующее свойство: когда система состоит из нескольких частей с отдельными энергиями, законы распределения фаз отдельных частей имеют одну и ту же природу. Это свойство чрезмерно упрощает рассмотрение вопросов термодинамики и является основой самых важных ее соотношений...

Когда множество систем распределено по фазе описанным выше способом, т.е. когда индекс вероятности (P) является линейной функцией энергии (E), мы можем сказать, что множество имеет *каноническое распределение*, и можем назвать делитель энергии (H) модулем распределения".

В литературе по физике уравнение (11.3) обычно называют каноническим распределением [877] или распределением Гиббса [611]. В литературе по нейронным сетям оно называется распределением Гиббса, распределением Больцмана, а также распределением Больцмана–Гиббса.

В контексте нейронных сетей, как предмета внимания этой книги, параметр T можно рассматривать как *псевдотемпературу*, управляющую термальными флуктуациями, представляющими эффект "синаптического шума" в нейроне. Таким образом, точная шкала в данных условиях не применима. Следовательно, константу k_B можно принять равной единице и переписать выражения для вероятности p_i и функции разбиения Z следующим образом:

$$p_i = \frac{1}{Z} \exp\left(-\frac{E_i}{T}\right) \tag{11.5}$$

и

$$Z = \sum_i \exp\left(-\frac{E_i}{T}\right). \tag{11.6}$$

Следовательно, толкование статистической механики будет основываться именно на этих двух определениях. При этом величину T будем называть просто *температурой системы* (temperature of the system). Из выражения (11.5) можно заметить, что величину $-\log\ p_i$ можно трактовать как одну из форм энергии, измеренную при единичной температуре.

Свободная энергия и энтропия

Свободная энергия физической системы по Гельмгольцу обозначается символом F и в терминах функции разбиения Z определяется следующим образом:

$$F = -T\ \log\ Z. \tag{11.7}$$

Соответственно *средняя энергия* системы имеет следующее значение:

$$\langle E \rangle = \sum_i p_i E_i, \tag{11.8}$$

где обозначение <·> применяется для операции усреднения по множеству. Таким образом, подставляя (11.5) в (11.8), видим, что разность между средней и свободной энергиями определяется следующей формулой:

$$\langle E \rangle - F = -T \sum_i p_i \log p_i. \tag{11.9}$$

Величина в правой части выражения (11.9), за исключением температуры T, является не чем иным, как *энтропией* системы:

$$H = -\sum_i p_i \log p_i. \tag{11.10}$$

Следовательно, выражение (11.9) можно переписать в следующем виде:

$$\langle E\rangle - F = TH,$$

или, эквивалентно,

$$F = \langle E\rangle - TH. \tag{11.11}$$

Теперь рассмотрим две системы, A и A', находящиеся друг с другом в термальном контакте. Предположим, что система A мала по сравнению с системой A'. В этом случае система A' выступает в роли теплового аккумулятора с некоторой постоянной температурой T. Общая энтропия этих двух систем будет стремиться к росту в соответствии с формулой

$$\Delta H + \Delta H' \geq 0,$$

где ΔH и $\Delta H'$ — изменения энтропии систем A и A' соответственно. В свете формулы (11.11) значение этого соотношения состоит в том, что свободная энергия системы F стремится к понижению до достижения минимума в состоянии равновесия. Из статистической механики известно, что общее распределение вероятности определяется распределением Гиббса. Также существует важный принцип, называемый *принципом минимума свободной энергии* (principle of minimal free energy), который формулируется следующим образом [611], [814].

Минимум свободной энергии в стохастической системе по переменным системы достигается в точке термального равновесия, определяемой распределением Гиббса.

Природа стремится найти физическую систему с минимальной свободной энергией.

11.3. Цепи Маркова

Рассмотрим систему, развитие которой определяется стохастическим процессом $\{X_n, n = 1, 2, \dots\}$, состоящим из семейства случайных переменных. Значение x_n, характеризующее случайную переменную X_n в дискретный момент времени n, называется *состоянием* системы в этот момент. Пространство всех возможных значений, которые могут принимать эти случайные переменные, называют *пространством состояний* (state space) системы. Если структура стохастического процесса $\{X_n, n = 1, 2, \dots\}$ такова, что условное распределение вероятности X_{n+1} зависит исключительно от предыдущего значения X_n и не зависит от всех более ранних значений, то такой

процесс называется *цепью Маркова*, или *Марковской цепью* (Markov chain) [71], [294]. Формально это можно записать так:

$$P(X_{n+1} = x_{n+1}|X_n = x_n, \ldots, X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n). \quad (11.12)$$

Это соотношение называется *свойством Маркова* (Markov property). Формально это свойство описывается следующим образом.

Последовательность случайных переменных $X_1, \ldots, X_n, X_{n+1}$ *принимает форму цепи Маркова, если вероятность нахождения системы в состоянии* x_{n+1} *в момент времени* $n+1$ *зависит исключительно от вероятности нахождения системы в момент времени* n *в состоянии* x_n.

Таким образом, цепи Маркова можно рассматривать как *порождающие модели* (generative model), состоящие из множества попарно связанных друг с другом состояний. Каждый раз, когда система переходит в конкретное состояние, именно с ним ассоциируется выход системы.

Вероятности перехода

В цепи Маркова переход системы из одного состояния в другое является *вероятностным* (probabilistic). Однако при этом выход системы является *детерминированным* (deterministic). Пусть *вероятность перехода* системы из состояния i (момент времени n) в состояние j (момент времени $n+1$) имеет следующий вид:

$$p_{ij} = P(X_{n+1} = j|X_n = i). \quad (11.13)$$

Так как величины p_{ij} являются условными вероятностями, они должны подчиняться двум условиям:

$$p_{ij} \geq 0(i, j) \quad (11.14)$$

и

$$\sum_j p_{ij} = 1 \text{ для всех } i. \quad (11.15)$$

Предполагается, что вероятности перехода фиксированы и не изменяются во времени. Это значит, что условие (11.13) удовлетворяется в любой момент времени n. Такая сеть Маркова называется *гомогенной* (homogeneous) по времени.

Если система имеет конечное множество возможных состояний K, вероятности перехода формируют матрицу размерности $K \times K$:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1K} \\ p_{21} & p_{22} & \cdots & p_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ p_{K1} & p_{K2} & \cdots & p_{KK} \end{bmatrix}. \tag{11.16}$$

Элементы этой матрицы удовлетворяют условиям (11.14) и (11.15). Последнее условие гарантирует равенство суммы всех элементов каждой из строк единице. Такая матрица называется *стохастической* (stochastic matrix). Любая стохастическая матрица может служить матрицей вероятностей перехода.

Определение одношаговой вероятности представлено формулой (11.13). Его можно обобщить для случая, когда переход из одного состояния в другое занимает несколько шагов. Обозначим *вероятность перехода за* m шагов из состояния i в состояние j символом $p_{ij}^{(m)}$:

$$p_{ij}^{(m)} = P(X_{n+m} = x_j | X_n = x_i), \quad m = 1, 2, \ldots. \tag{11.17}$$

Величину $p_{ij}^{(m)}$ можно рассматривать как сумму всех промежуточных состояний k, через которые система проходит по пути из i в j. В частности, величина $p_{ij}^{(m+1)}$ связана с величиной $p_{ij}^{(m)}$ следующим рекурсивным соотношением:

$$p_{ij}^{(m+1)} = \sum_k p_{ik}^{(m)} p_{kj}, \quad m = 1, 2, \ldots, \tag{11.18}$$

где

$$p_{ik}^{(1)} = p_{ik}.$$

Равенство (11.18) можно обобщить следующим образом:

$$p_{ij}^{(m+n)} = \sum_k p_{ik}^{(m)} p_{kj}^{(n)}, \quad (m, n) = 1, 2, \ldots. \tag{11.19}$$

Это частный случай *равенства Чепмена–Колмогорова* (Chapmen-Kolmogorov identity) [294].

Если состояние цепи может возникать через интервалы времени, кратные d, где d — наибольшее из таких целых чисел, считается, что такое состояние имеет *периодичность* d. Цепь Маркова называется *апериодичной* (aperiodic), если все ее состояния имеют периодичность 1.

Свойства рекуррентности

Предположим, что цепь Маркова начинается с состояния i. Состояние i называется *рекуррентным* (recurrent), если цепь Маркова возвращается в состояние i с вероятностью 1:

$$f_i = P(\text{возвращения в состояние } i) = 1.$$

Если вероятность f_i меньше единицы, такое состояние i называется *транзитным* (transient) [633].

Если цепь Маркова начинается с рекуррентного состояния, это состояние повторяется бесконечное число раз. Если же цепь Маркова начинается с транзитного состояния, это состояние может повториться только конечное число раз. Это можно объяснить следующим образом. Повторения состояния i можно рассматривать как *пробы Бернулли* (Bernoulli trial) с вероятностью успеха f_i. Таким образом, количество возвратов можно представить как геометрическую случайную переменную со средним значением $(1 - f_i^{-1})$. Если f_i меньше единицы, следовательно, количество бесконечных успешных попыток равно нулю. Таким образом, транзитное состояние не повторяется после конечного числа возвратов [633].

Если цепь Маркова состоит из нескольких транзитных и нескольких рекуррентных состояний, то процесс может перемещаться только между рекуррентными состояниями.

Несократимые цепи Маркова

Состояние j цепи Маркова называется *достижимым* (accessible) из состояния i, если с положительной вероятностью существует конечная последовательность переходов из i в j. Если состояния i и j достижимы друг из друга, считается, что они *связаны* друг с другом. Эта связь обозначается следующим выражением: $i \leftrightarrow j$. Естественно, если состояние i связано с состоянием j, а последнее, в свою очередь, с состоянием k, то состояния i и k также являются связанными.

Если два состояния цепи Маркова связаны друг с другом, они считаются принадлежащими одному *классу*. В общем случае состояния цепи Маркова принадлежат одному или нескольким несвязанным классам. Если же состояния принадлежат всего одному классу, такая цепь называется *несократимой* (irreducible) или *неразложимой* (indecomposible). Другими словами, начиная с любого состояния несократимой цепи Маркова, можно достичь любого другого ее состояния с положительной вероятностью. Для большинства приложений *сократимые* цепи представляют малый интерес. Поэтому мы ограничимся рассмотрением только несократимых цепей.

Пусть несократимая цепь Маркова начинается с некоторого рекуррентного состояния i в момент времени $n = 0$. Пусть $T_i(k)$ — время, которое прошло между

$(k-1)$ и k-м возвратом в состояние i. *Среднее время возврата* (mean recurrence time) определяется как ожидание $T_i(k)$ по всем возвратам k. *Установившимся значением вероятности* (steady-state probability) состояния i называется величина, обратная к среднему времени возврата $E[T_i(k)]$:

$$\pi_i = \frac{1}{E[T_i(k)]}.$$

Если $E[T_i(k)] < \infty$ и соответственно $\pi_i > 0$, то состояние i называется *положительно рекуррентным* (positive recurrent). В противном случае оно называется *состоянием с нулевой рекуррентностью* (null recurrent state). Последний случай трактуется следующим образом: цепь Маркова случайно достигает такой точки, из которой возврат в состояние i невозможен. Положительная и нулевая рекуррентность являются *различными* свойствами класса. Это значит, что цепи Маркова с одновременной положительной и нулевой рекуррентностью состояний являются сократимыми.

Эргодические цепи Маркова

Под термином *эргодичность* (ergodicity) понимается возможность подстановки среднего по времени вместо среднего по множеству. В контексте цепей Маркова при этом подразумевается, что соотношение времени, проведенного цепью в состоянии i, соответствует установившемуся значению вероятности π_i. Это можно объяснить следующим образом. *Соотношение времени, проведенного в состоянии* i после k *возвратов*, определяется следующим образом:

$$v_i(k) = \frac{k}{\sum_{l=1}^{k} T_i(l)}.$$

Время возврата $T_i(l)$ формирует последовательность независимо и равномерно распределенных случайных переменных, так как по определению время каждого из возвратов статистически независимо от всех предыдущих времен возвратов. Более того, в случае рекуррентного состояния i цепь возвращается в это состояние бесконечное число раз. Исходя из этого, при достижении количеством возвратов k бесконечности, согласно *закону больших чисел*, пропорция времени, проведенная в состоянии i, достигает установившегося значения вероятности, т.е.

$$\lim_{k\to\infty} v_i(k) = \pi_i \text{ для } i = 1, 2, \ldots, K. \tag{11.20}$$

Достаточным (но не необходимым) условием эргодичности цепей Маркова является их одновременная апериодичность и несократимость.

Сходимость к стационарным распределениям

Рассмотрим эргодическую цепь Маркова, характеризуемую стохастической матрицей $\mathbf{P}$. Пусть вектор-строка $\boldsymbol{\pi}^{(n-1)}$ определяет *вектор распределения состояний* (state distribution vector) цепи в момент времени $n-1$. Его j-й элемент является вероятностью, с которой в момент времени $n-1$ цепь находится в состоянии x_j. В момент времени n этот вектор распределения состояний определяется следующим образом:

$$\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}^{(n-1)}\mathbf{P}. \tag{11.21}$$

Итеративно используя выражение (11.21), получим:

$$\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}^{(n-1)}\mathbf{P} = \boldsymbol{\pi}^{(n-2)}\mathbf{P}^2 = \boldsymbol{\pi}^{(n-3)}\mathbf{P}^3 = \ldots,$$

что в конечном счете приводит к следующей формуле:

$$\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}^{(0)}\mathbf{P}^n, \tag{11.22}$$

где $\boldsymbol{\pi}^{(0)}$ — *начальное значение* вектора распределения состояний. Другими словами, вектор распределения состояний цепи Маркова в момент времени n является произведением исходного значения этого вектора на n-ю степень стохастической матрицы $\mathbf{P}$.

Обозначим символом $p_{ij}^{(n)}$ ij-й элемент матрицы $\mathbf{P}$. Предположим, что при устремлении n к бесконечности вероятность $p_{ij}^{(n)}$ стремится к некоторому значению π_j, не зависящему от i, где π_j — стационарная вероятность состояния j. Соответственно при больших n матрица $\mathbf{P}^n$ достигает своей предельной формы квадратной матрицы с идентичными строками:

$$\lim_{n\to\infty}\mathbf{P}^n = \begin{bmatrix} \pi_1 & \pi_2 & \ldots & \pi_K \\ \pi_1 & \pi_2 & \ldots & \pi_K \\ \ldots & \ldots & \ldots & \ldots \\ \pi_1 & \pi_2 & \ldots & \pi_K \end{bmatrix} = \begin{bmatrix} \boldsymbol{\pi} \\ \boldsymbol{\pi} \\ \ldots \\ \boldsymbol{\pi} \end{bmatrix}, \tag{11.23}$$

где $\boldsymbol{\pi}$ — вектор-строка, состоящий из элементов $\pi_1, \pi_2, \ldots, \pi_K$. После перестановки слагаемых из выражения (11.22) получим следующую формулу:

$$\left[\sum_{j=1}^{K}\pi_j^{(0)} - 1\right]\boldsymbol{\pi} = \mathbf{0}.$$

Так как по определению $\sum_{j=1}^{K}\pi_j^{(0)} = 1$, вектор $\boldsymbol{\pi}$ не будет зависеть от начального распределения.

Теперь можно сформулировать *теорему об эргодичности* (ergodicity theorem) для цепей Маркова [71], [294]:

Пусть эргодическая цепь Маркова с состояниями $x_1, x_2, \ldots, x_K$ и стохастической матрицей $\mathbf{P}= \{p_{ij}\}$ является несократимой. Тогда эта цепь имеет единственное стационарное распределение, к которому она сходится из любого начального состояния. Это значит, что существует единственный набор чисел $\{\pi_j\}_{j=1}^K$, такой, что

$$\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j \quad \textit{для всех } i, \tag{11.24}$$

$$\pi_i > 0 \quad \textit{для всех } j, \tag{11.25}$$

$$\sum_{j=1}^{K} \pi_j = 1, \tag{11.26}$$

$$\pi_j = \sum_{i=1}^{K} \pi_i p_{ij} \quad \textit{для всех } j = 1, 2, \ldots, K. \tag{11.27}$$

И наоборот, предположим, что цепь Маркова является несократимой и апериодической, а также существуют числа $\{\pi_j\}_{j=1}^K$, удовлетворяющие условиям (11.25)–(11.27). Тогда эта цепь является эргодической, величины π_j вычисляются по формуле (11.24), а среднее время возврата в состояние j равно $1/\pi_j$.

Распределение вероятности $\{\pi_j\}_{j=1}^K$ называется *инвариантным* или *стационарным*. Такое название объясняется тем, что по достижении такого распределения оно сохраняется. В свете теоремы об эргодичности можно утверждать следующее.

- Начиная с любого исходного состояния, вероятности перехода в цепи Маркова сходятся к стационарному распределению, если такое распределение существует.
- Если цепь эргодическая, то стационарное распределение в цепи Маркова полностью не зависит от ее начального распределения.

Пример 11.1

Рассмотрим цепь Маркова с *диаграммой перехода состояний*, показанной на рис. 11.1. Эта цепь имеет два состояния — x_1 и x_2. Стохастическая матрица этой цепи имеет следующий вид:

$$\mathbf{P} = \begin{bmatrix} 1/4 & 3/4 \\ 1/2 & 1/2 \end{bmatrix}.$$

Эта матрица удовлетворяет условиям (11.14) и (11.15).

Предположим, что начальным состоянием является следующее:

$$\boldsymbol{\pi}^{(0)} = [1/6, 5/6].$$

По формуле (11.21) получим следующий вектор распределения состояний для момента времени $n = 1$:

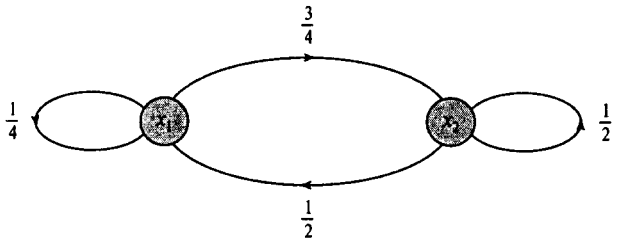


Рис. 11.1. Диаграмма перехода состояний в цепи Маркова для примера 11.1

$$\boldsymbol{\pi}^{(1)} = \boldsymbol{\pi}^{(0)}\mathbf{P} = \begin{bmatrix} 1/6 & 5/6 \end{bmatrix} \begin{bmatrix} 1/4 & 3/4 \\ 1/2 & 1/2 \end{bmatrix} = \begin{bmatrix} 11/24 & 13/24 \end{bmatrix}.$$

Возводя стохастическую матрицу $\mathbf{P}$ в степени 2, 3 и 4, получим:

$$\mathbf{P}^2 = \begin{bmatrix} 0.4375 & 0.5625 \\ 0.3750 & 0.6250 \end{bmatrix},$$

$$\mathbf{P}^3 = \begin{bmatrix} 0.4001 & 0.5999 \\ 0.3999 & 0.6001 \end{bmatrix},$$

$$\mathbf{P}^4 = \begin{bmatrix} 0.4000 & 0.6000 \\ 0.4000 & 0.6000 \end{bmatrix}.$$

Таким образом, $\pi_1 = 0,4000$, а $\pi_2 = 0,6000$. В данном примере сходимость к стационарному состоянию была обеспечена всего за 4 итерации. Так как обе величины, π_1 и π_2, больше нуля, оба состояния являются положительно рекуррентными, а сеть — несократимой. Также заметим, что данная цепь является апериодической, так как наибольший общий делитель всех чисел $n \geq 1$, таких, что $(\mathbf{P}^n)_{ij} > 0$, равен единице. Исходя из сказанного, делаем вывод о том, что цепь Маркова на рис. 11.1 является эргодической. ■

Пример 11.2

Рассмотрим цепь Маркова, имеющую стохастическую матрицу, содержащую отдельные нулевые элементы:

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 1 \\ 1/3 & 1/6 & 1/2 \\ 3/4 & 1/4 & 0 \end{bmatrix}.$$

Диаграмма перехода состояний этой цепи представлена на рис. 11.2.

Применяя выражение (11.27), получим следующую систему уравнений:

$$\begin{cases} \pi_1 = \frac{1}{3}\pi_2 + \frac{3}{4}\pi_3, \\ \pi_2 = \frac{1}{6}\pi_2 + \frac{1}{4}\pi_3, \\ \pi_3 = \pi_1 + \frac{1}{2}\pi_2. \end{cases}$$

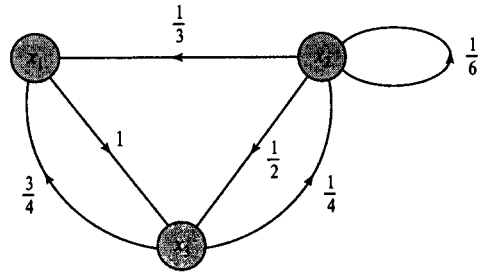


Рис. 11.2. Диаграмма перехода состояний для сети Маркова из примера 11.2

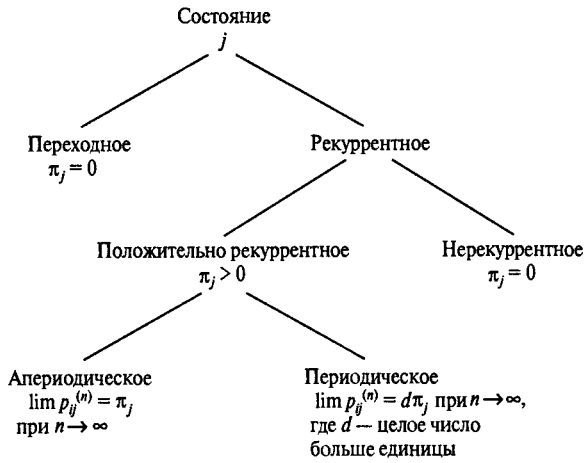


Рис. 11.3. Классификация состояний цепи Маркова и связанного с ними поведения

Решив эту систему уравнений, находим следующие значения:

$$\pi_1 = 0,3953,$$

$$\pi_2 = 0,1395,$$

$$\pi_3 = 0,4652.$$

Итак, даиная цепь Маркова является эргодической со стационарным распределеиием, определяемым величинами π_1, π_2 и π_3. ■

Классификация состояний

На основе представленного материала можно создать диаграмму классов, к которым может принадлежать состояние (рис. 11.3) [294], [633].

Принцип детального баланса

Выражения (11.25) и (11.26) подчеркивают тот факт, что числа π_i являются вероятностями. Выражение (11.27) также является критичным, так как выполнение этого

условия требуется для того, чтобы цепь Маркова была несократимой и, таким образом, существовало стационарное распределение. Это последнее условие является записью *принципа детального баланса* (principle of detailed balance), известного в кинетике первого порядка. Этот принцип утверждает, что в состоянии термального равновесия частота возникновения любого перехода равна соответствующей частоте обратного перехода [877]:

$$\pi_i p_{ij} = \pi_j p_{ji}. \tag{11.28}$$

Для вывода соотношения (11.27) можно переставить порядок суммирования в правой части уравнения:

$$\sum_{i=1}^{K} \pi_i p_{ij} = \sum_{i=1}^{K} \left(\frac{\pi_i}{\pi_j} p_{ij} \right) \pi_j = \sum_{i=1}^{K} (p_{ji}) \pi_j = \pi_j.$$

Во второй строке этого выражения использовался принцип детального баланса, а в последней — тот факт, что вероятности перехода в цепи Маркова удовлетворяют следующему условию (см. (11.15), где роли i и j поменялись местами):

$$\sum_{i=1}^{K} p_{ji} = 1 \text{ для всех } j.$$

Обратите внимание, что принцип детального баланса предполагает, что распределение $\{\pi_j\}$ является стационарным.

11.4. Алгоритм Метрополиса

Теперь, когда стало понятно строение цепей Маркова, их можно использовать для вывода стохастического алгоритма моделирования развития физических систем до состояния термального равновесия. Этот алгоритм получил имя своего создателя – *Метрополиса* (Metropolis algorithm) [730]. Он представляет собой модифицированный алгоритм Монте-Карло, введенный еще в раннюю эпоху моделирования достижения температурного равновесия стохастическими системами, состоящими из атомов.

Предположим, что случайная переменная X_n, представляющая произвольную цепь Маркова, в момент времени n находится в состоянии x_i. Сгенерируем случайным образом новое состояние x_j, представляющее собой реализацию другой случайной переменной Y_n. Предполагается, что генерация этого нового состояния удовлетворяет условию симметрии:

$$P(Y_n = x_j | X_n = x_i) = P(Y_n = x_i | X_n = x_j).$$

Обозначим символом ΔE разницу в энергии, возникающую при переходе системы из состояния $X_n = x_i$ в состояние $Y_n = x_j$. Если эта величина отрицательна, то переход приводит к состоянию с меньшей энергией и такой переход принимается. После этого новое состояние считается точкой старта следующего шага алгоритма (т.е. мы принимаем $X_{n+1} = Y_n$). Если же разница в энергии положительна, алгоритм продолжает работу по вероятностной схеме в этой же точке. Выбираем случайное число ξ, равномерно распределенное в диапазоне [0, 1]. Если $\xi < \exp(-\Delta E/T)$, где T — рабочая температура, переход принимается, при этом считается, что $X_{n+1} = Y_n$. В противном случае переход отвергается, при этом полагается, что $X_{n+1} = X_n$ (т.е. на следующем шаге алгоритма используется старая конфигурация).

Выбор вероятности перехода

Пусть цепь Маркова имеет априорные вероятности перехода τ_{ij}, удовлетворяющие следующим трем условиям.

1. *Неотрицательность*:

$$\tau_{ij} \geq 0 \text{ для всех } (i, j).$$

2. *Нормализация*:

$$\sum_j \tau_{ij} = 1 \quad \text{для всех } i.$$

3. *Симметрия*:

$$\tau_{ij} = \tau_{ji} \text{ для всех } (i, j).$$

Пусть π_i означает установившееся значение вероятности, в котором находится цепь Маркова в состоянии $x_i, i = 1, 2, \ldots, K$. Тогда симметрию τ_{ij} и частное π_j/π_i можно использовать для определения требуемого множества вероятностей переходов [115]:

$$p_{ij} = \begin{cases} \tau_{ij}\left(\frac{\pi_j}{\pi_i}\right) & \text{для } \frac{\pi_j}{\pi_i} < 1, \\ \tau_{ij} & \text{для } \frac{\pi_j}{\pi_i} \geq 1. \end{cases} \tag{11.29}$$

Для обеспечения нормализации вероятностей перехода к значению единицы введем следующее определение вероятности отсутствия перехода:

$$p_{ii} = \tau_{ii} + \sum_{j \neq i} \tau_{ij} \left(1 - \frac{\pi_j}{\pi_i}\right) = 1 - \sum_{j \neq i} \alpha_{ij} \tau_{ij}, \tag{11.30}$$

где α_{ij} — *вероятность перехода*, определенная как

$$\alpha_{ij} = \min\left(1, \frac{\pi_j}{\pi_i}\right). \tag{11.31}$$

Теперь возникает один важный вопрос: как выбрать частное π_j/π_i? Для того чтобы ответить на этот вопрос, следует выбрать такое распределение вероятностей, которое требуется для перехода распределения вероятности цепи Маркова в распределение Гиббса:

$$\pi_j = \frac{1}{Z} \exp\left(-\frac{E_j}{T}\right).$$

В этом случае отношение вероятностей π_j/π_i принимает простую форму:

$$\frac{\pi_j}{\pi_i} = \exp\left(-\frac{\Delta E}{T}\right), \tag{11.32}$$

где

$$\Delta E = E_j - E_i. \tag{11.33}$$

Используя частное распределений вероятности, можно избежать зависимости от функции разбиения Z. По своему определению вероятности перехода являются неотрицательными числами, нормализованными к единице (см. (11.14) и (11.15)). Более того, они удовлетворяют принципу детального баланса, определенного выражением (11.28). Этот принцип является достаточным условием температурного равновесия. Чтобы продемонстрировать, как выполняется принцип детального баланса, рассмотрим следующие случаи.

Случай 1. $\Delta E < 0$

Предположим, что при переходе из состояния x_i в состояние x_j произошло отрицательное изменение энергии. Из выражения (11.32) получим π_j/π_i>1, после чего с помощью (11.29) получим:

$$\pi_i p_{ij} = \pi_i \tau_{ij} = \pi_i \tau_{ji}$$

и

$$\pi_j p_{ji} = \pi_j \left(\frac{\pi_i}{\pi_j} \tau_{ji} \right) = \pi_i \tau_{ji}.$$

Исходя из этого, принцип детального баланса удовлетворяется при $\Delta E<0$.

Случай 2. $\Delta E > 0$

Теперь предположим, что при переходе из состояния x_i в x_j произошло положительное изменение энергии. В этом случае $\pi_j/\pi_i<1$, откуда с помощью выражения (11.29) получим:

$$\pi_i p_{ij} = \pi_i \left(\frac{\pi_j}{\pi_i} \tau_{ij} \right) = \pi_j \tau_{ij} = \pi_j \tau_{ji}$$

и

$$\pi_j p_{ji} = \pi_j \tau_{ji}.$$

При этом снова удовлетворяется принцип детального баланса.

Чтобы получить общую картину, давайте проясним использование *априорных* вероятностей перехода, которые обозначены символом τ_{ij}. Эти вероятности перехода на самом деле являются вероятностными моделями случайного шага в алгоритме Метрополиса. Из ранее представленного описания этого алгоритма известно, что за случайным шагом следует случайное решение. Отсюда можно заключить, что вероятности перехода p_{ij}, определенные по формулам (11.29) и (11.30) в терминах *априорных* вероятностей перехода τ_{ij} и стационарных вероятностей π_i, на самом деле являются подходящими для алгоритма Метрополиса.

Важно отметить, что стационарное распределение, генерируемое алгоритмом Метрополиса, не однозначно определяет цепь Маркова. Распределение Гиббса в состоянии равновесия может быть сгенерировано с использованием правила, отличного от метода Монте-Карло, применяемого в алгоритме Метрополиса. Например, его можно сгенерировать с помощью правила обучения Больцмана [9]. Это правило мы рассмотрим в разделе 11.7.

11.5. Метод моделирования отжига

Теперь займемся поиском системы с низкой энергией, состояния которой упорядочены в цепь Маркова. Из выражения (11.11) видно, что при достижении температурой T нуля свободная энергия системы F достигает среднего значения энергии $\langle E \rangle$. При $F \to \langle E \rangle$ из принципа минимума свободной энергии видно, что распределение Гиббса, являющееся стационарным распределением в цепи Маркова, разрушается в точке глобального минимума средней энергии при $T \to 0$. Другими словами, упо-

рядоченные состояния с низкой энергией более предпочтительны при низких температурах. Все эти наблюдения приводят к следующему вопросу: "Почему нельзя напрямую применить алгоритм Метрополиса для генерации популяций представителей конфигурации стохастических систем при очень низких температурах?" Автор не советует использовать эту стратегию, так как скорость сходимости цепи Маркова к точке термального равновесия при малых температурах чрезвычайно низка. Вместо этого для достижения вычислительной эффективности рекомендуется работать со стохастическими системами при высоких температурах, где скорость сходимости к точке равновесия велика, после чего работать уже в точке равновесия с системой при более низких температурах. Таким образом, будем использовать комбинацию двух связанных составляющих.

- Расписание, определяющее уровень, на котором температуру следует понизить.
- Алгоритм, подобный алгоритму Метрополиса, итеративно определяющий равновесное распределение для каждой новой температуры из расписания, используя конечное состояние системы при предыдущей температуре в качестве точки начала работы с более низкой температурой.

Описанная двухшаговая схема лежит в основе широко используемого метода *стохастической релаксации* (stochastic relaxation), называемого *моделированием отжига* (simulated annealing)[2] [560]. Свое название этот подход получил по аналогии с процессами в физике или химии, где процесс начинается при более высокой температуре, которая затем понижается до достижения точки температурного равновесия.

---

[2] Идея введения температуры и моделирования отжига в задачах комбинаторной оптимизации появилась независимо в [179] и [560].

В контексте физики отжиг является очень тонким процессом. В [560] обсуждалось поиятие "плавки" твердого тела, которое включало в себя поднятие температуры до максимального значения, при которой все части этого тела переходили в жидкую фазу, т.е. размещались случайным образом. После этого температура понижалась, позволяя всем частям организоваться в низкоэнергетическом состоянии "земли" соответствующей решетки. Если охлаждение происходило слишком быстро (т.е. у твердого тела не хватало времени для достижения термального равновесия на каждом температурном значении), полученный кристалл имел массу дефектов или субстанция могла сформировать стекло с отсутствием кристаллического порядка и метаустойчивой локально оптимальной структурой.

Понятие "отжига" хорошо иллюстрируется процессом изготовления стекла и является в некотором контексте задачей комбиниаторной оптимизации. Однако оно может ввести в заблуждение при рассмотрении многих других областей применения [115]. Например, если поднять "температуру" при обработке изображений до такого уровня, что все части организуются случайным образом, мы попросту потеряем само изображение – оно станет равномерно серым. В соответствующем контексте металлургии при отжиге железа или меди необходимо сохранять температуру отжига ниже температуры отжига металла, в противном случае заготовка будет разрушена.

Существует ряд важных параметров, характерных для процесса отжига в металлургии.

Температура отжига (annealing temperature) — температура, до которой нагревается металл или сплав.

Время отжига (annealing time) — продолжительность времени, в течение которого поддерживается повышенная температура.

Расписание охлаждения (cooling schedule) — определяет скорость понижения температуры.

Эти параметры имеют свои аналоги в моделировании отжига, что и будет описано в соответствующем подразделе.

Основной целью моделирования отжига является поиск глобального минимума функции стоимости, которая характеризует большую и сложную систему[3]. Этот метод является мощным инструментом решения невыпуклых задач оптимизации. Он объясняется одной достаточно простой идеей.

При оптимизации очень больших и сложных систем (т.е. систем с множеством степеней свободы) вместо постоянного движения вниз по склону старайтесь двигаться по склону вниз большую часть времени.

Моделирование отжига отличается от обычных итеративных алгоритмов оптимизации двумя важными аспектами.

- Этот алгоритм не "стопорится", так как выход из точки локального минимума всегда возможен при работе системы в условиях ненулевой температуры.
- Моделирование отжига является *адаптивным*. Большая часть основных свойств конечного состояния системы уже проявляются при высоких температурах, в то время как при более низких их состояние только уточняется.

Расписание отжига

Как уже говорилось ранее, алгоритм Метрополиса является основой процесса моделирования отжига, направленного на медленное понижение температуры T. Это значит, что сама температура T выступает в роли параметра *управления*. Процесс моделирования отжига будет сходиться к конфигурации с минимальной энергией при условии, что температура будет понижаться не быстрее, чем логарифмически. К сожалению, такое расписание отжига чрезвычайно медленно — настолько медленно, что вряд ли применимо на практике. На практике необходимо прибегнуть к *аппроксимации на конечном интервале времени* (finite-time approximation). Однако за это придется заплатить большую цену — алгоритм не будет гарантированно сходиться к точке глобального минимума с вероятностью 1. Тем не менее получаемая приближенная форма в большинстве практических приложений этого алгоритма способна привести к точке, достаточно близкой к оптимальному решению.

Для того чтобы реализовать приближение на конечном интервале времени алгоритма моделирования отжига, необходимо определить множество параметров, управляющих его сходимостью. Эти параметры объединяются в так называемое *расписание отжига* (annealing schedule) или *расписание охлаждения* (cooling schedule). Это расписание представляет собой конечную последовательность значений температу-

---

[3] Уравнение Ланжевина (Langevin) (с температурой, зависящей от времени) является основой еще одного важного алгоритма глобальной оптимизации, который был предложен в [384] и впоследствии проанализирован в [350]. Уравнение Ланжевина $\frac{\partial v(t)}{\partial t} = -\gamma v(t) + \Gamma(t)$, где $v(t)$ — скорость частиц с массой m, погруженных в жидкость; γ — константа, равная частному от деления коэффициента трения на массу m; $\Gamma(t)$ — удельная сила колебаний (на единицу массы). Уравнение Ланжевина было первым математическим соотношением, которое описывало неравновесную термодинамику.

ры и конечное число пробных переходов для каждой из этих температур. В [560] интересующие параметры определяются следующим образом[4].

- *Начальное значение температуры*. Начальное значение T_0 выбирается достаточно высоким для того, чтобы для алгоритма моделирования отжига были доступны все предлагаемые переходы.
- *Понижение температуры*. Обычно охлаждение осуществляется *экспоненциально*, а изменения, применяемые к температуре, являются небольшими. В частности, *функцию убывания* (decrement function) можно определить следующим образом:

$$T_k = \alpha T_{k-1}, k = 1, 2, \ldots, \tag{11.34}$$

где α — константа, меньшая единицы, но достаточно близкая к последней. Как правило, значения этой константы выбираются в диапазоне от 0,8 до 0,99. При каждой из температур предпринимается достаточно много попыток перехода, так чтобы среднее их количество не опускалось ниже 10 *принятых переходов* (accepted transition).

- *Конечное значение температуры*. Система постепенно охлаждается и отжиг останавливается, когда требуемое количество принятых переходов не достигается при трех последовательных температурах.

Последний критерий можно переопределить следующим образом: *коэффициент пропускания* (acceptance ratio), определяемый как количество принятых переходов, деленное на количество предложенных переходов, становится меньше некоторого наперед заданного значения [516].

Моделирование отжига для комбинаторной оптимизации

Моделирование отжига хорошо подходит, в частности, для решения задач комбинаторной оптимизации. Целью *комбинаторной оптимизации* (combinatorial optimization) является минимизация функции стоимости конечной дискретной системы, характеризуемой большим количеством возможных решений. В своей основе моделирование отжига использует алгоритм Метрополиса для генерации последовательности решений, с привлечением аналогии между физическими *многочастичными* (many-particle) системами и задачей комбинаторной оптимизации.

---

[4] Более сложные и теоретически обоснованные расписания отжига описаны в [1], [1079].

ТАБЛИЦА 11.1. Соответствия между терминологией статистической физики и комбинаторной оптимизации

Статистическая физика	*Комбинаторная оптимизация*
Пример (sample)	Экземпляр задачи (problem instance)
Состояние (state), конфигурация	Конфигурация (configuration)
Энергия (energy)	Функция стоимости (cost function)
Температура	Параметр управления
Энергия заземления (ground-state energy)	Минимальная стоимость
Конфигурация заземления (ground-state configuration)	Оптимальная конфигурация (optimal configuration)

При моделировании отжига энергия E_i в распределении Гиббса (11.5) интерпретируется как числовая стоимость, а температура T — как параметр управления. Эта числовая стоимость ставит в соответствие каждой из конфигураций задачи комбинаторной оптимизации некоторое скалярное значение, которое отражает приемлемость данной конфигурации для решения. Следующий вопрос моделирования отжига сводится к следующему: как идентифицировать конфигурации и локально генерировать новые конфигурации из уже имеющихся. Именно в этом вопросе алгоритм Метрополиса играет особо важную роль. В табл. 11.1 приведены соответствия между терминологией статистической физики и комбинаторной оптимизации [115].

11.6. Распределение Гиббса

Подобно алгоритму Метрополиса, *квантизатор Гиббса*[5] (Gibbs sampler) генерирует цепь Маркова с распределением Гиббса, выступающим в роли равновесного распределения. Однако вероятности перехода, связанные с квантизатором Гиббса, не являются стационарными [343]. При окончательном анализе выбор между квантизатором Гиббса и алгоритмом Метрополиса зависит от технических деталей рассматриваемой задачи.

Для того чтобы приступить к описанию этой схемы квантования, рассмотрим K-мерный случайный вектор $\mathbf{X}$, состоящий из компонентов $X_1, X_2, \ldots, X_K$. Предположим, что известно условное распределение X_k при заданных значениях всех

[5] Квантованне Гиббса связано со статистнческой фнзнкой и широко используется в обработке изображений, нейронных сетях и статистике, что последовало за его формальным описанием в [342], [343]. В последней работе также рассматрнваются другие подходы к квантованию, основанные на нных способах получения числовых оценок граничных распределений вероятностн. В [431] было представлено обобщение алгоритма Метрополиса, относительно которого квантование Гиббса является частным случаем, и был показан его потенцнал в решении числовых задач статистики.

остальных компонентов вектора $\mathbf{X}$ ($k = 1, 2, \ldots, K$). Ставится задача получения числовой оценки *граничной плотности* (marginal density) случайной переменной X_k для всех k. Квантизатор Гиббса генерирует значения для условного распределения каждого из компонентов вектора $\mathbf{X}$ при заданных значениях всех остальных его компонентов. В частности, начиная с произвольной конфигурации $\{x_1(0), x_2(0), \ldots, x_K(0)\}$, первой итерации квантования Гиббса создаются следующие подвыборки:

$x_1(1)$ выбирается из распределения X_1 при известных $x_2(0)$, $x_3(0)$, ...,$x_K(0)$.

$x_2(1)$ выбирается из распределения X_2 при известных $x_1(1)$, $x_3(0)$, ...,$x_K(0)$.

...

$x_k(1)$ выбирается из распределения X_k при известных $x_1(1)$, $x_{k-1}(1)$, $x_{k+1}(0)$,...,$x_K(0)$.

...

$x_K(1)$ выбирается из распределения X_K при известных $x_1(1)$, $x_2(1)$, ...,$x_{K-1}(1)$.

Аналогичные действия выполняются на следующих итерациях схемы квантования. При этом следует особо отметить следующие вопросы.

1. Каждый из компонентов случайного вектора $\mathbf{X}$ "посещается" в естественном порядке. В результате на каждой итерации формируется K случайных величин.
2. Новое значение компонента X_{k-1} используется сразу же после создания, при формировании случайного значения X_k, k =2, 3, ..., K.

Из представленного материала видно, что квантизатор Гиббса является *итеративной адаптивной схемой* (iterative adaptive scheme). После n итераций получается K случайных значений $X_1(n), X_2(n),\ldots,X_K(n)$. При относительно мягких условиях для квантизатора Гиббса выполняются следующие теоремы [342], [343].

1. *Теорема о сходимости. Случайная переменная $X_k(n)$ сходится по распределению к истинному распределению вероятности X_k для $k = 1, 2, \ldots, K$ при стремлении n к бесконечности, т.е.*

$$\lim_{n\to\infty} P(X_k^{(n)} \le x | x_k(0)) = F_{X_k}(x), \quad k = 1, 2, \ldots, K, \tag{11.35}$$

где $F_{Xk}(x)$ — функция граничного распределения вероятности X_k.

В [343] был доказан более строгий результат. В частности, в ней не требовалось посещение каждого компонента случайного вектора $\mathbf{X}$ в естественном порядке. Сходимость квантования Гиббса сохраняется и при произвольном порядке посещения компонентов $\mathbf{X}$. Это значит, что эта схема не зависит от значений переменных и что каждый из компонентов $\mathbf{X}$ посещается "бесконечно часто".

2. *Теорема о скорости сходимости. Совместное распределение вероятности случайных переменных $X_1(n), X_2(n), \ldots, X_K(n)$ сходится к истинному совместному распределению вероятности $X_1, X_2, \ldots, X_K$ пропорционально n.*

В теореме предполагается, что компоненты вектора **X** посещаются в естественном порядке. Однако если используется произвольное, но бесконечно частое посещение, скорость сходимости претерпевает незначительное изменение.

3. *Теорема об эргодичности. Для любой измеримой функциид от (к примеру) случайных переменных* $X_1, X_2, \ldots, X_K$, *ожидания которой известны, с вероятностью 1 выполняется:*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} g(X_1(i), X_2(i), \ldots, X_K(i)) \rightarrow E[g(X_1, X_2, \ldots, X_K)]. \qquad (11.36)$$

Теорема об эргодичности предлагает способ использования выхода квантователя Гиббса для вычисления числовых оценок искомых граничных плотностей.

Квантователь Гиббса используется в машине Больцмана для квантования на основе распределений скрытых нейронов. Эта стохастическая машина рассматривается в следующем разделе. В контексте стохастических машин, использующих двоичные элементы (т.е. машин Больцмана), следует заметить, что квантователь Гиббса является вариантом алгоритма Метрополиса. В стандартной форме алгоритма Метрополиса движение происходит вниз по склону с вероятностью 1. В противоположность этому в альтернативной форме алгоритма Метрополиса мы движемся вниз по склону с вероятностью единица минус экпоненциал *дефицита энергии* (energy gap) (т.е. вклад в правило движения вверх по склону). Другими словами, если изменение энергии E произошло в сторону уменьшения или энергия осталась неизменной, то изменение принимается. Если изменение предполагает увеличение энергии, оно принимается с вероятностью $\exp(-\Delta E)$, в противном случае оно отвергается, сохраняя предыдущее состояние [777].

11.7. Машина Больцмана

Машина Больцмана (Boltzmann machine) представляет собой стохастическую машину, компонентами которой являются стохастические нейроны. *Стохастический нейрон* находится в одном из двух возможных вероятностных состояний. Этим двум состояниям формально можно присвоить значения $+1$ (соответствующее включенному состоянию) и -1 (соответствующее выключенному состоянию). Аналогично, можно принять значениями этих состояний $+1$ и 0 соответственно. Примем первое допущение. Еще одним отличительным свойством машины Больцмана является использование *симметричных синаптических связей* (symmetric synaptic connection) между нейронами. Использование этой формы синаптической связи обусловлено соглашениями статистической физики.

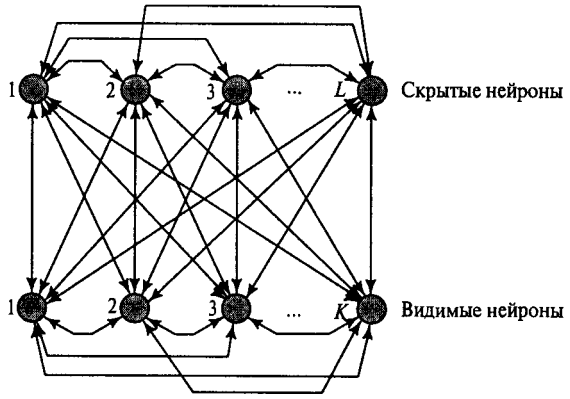


Рис. 11.4. Архитектурный граф машины Больцмана (где K — количество видимых, а L — количество скрытых нейронов)

Стохастические нейроны машины Больцмана разбиваются на две функциональные группы: *видимые* (visible) и *скрытые* (hidden) (рис. 11.4). Видимые нейроны[6] предоставляют интерфейс между сетью и средой, в которой она работает. Во время этапа обучения сети видимые нейроны фиксируются в своих специфичных состояниях, определяемых средой. С другой стороны, скрытые нейроны всегда работают свободно — они используются для выражения ограничений, содержащихся во входных векторах. Скрытые нейроны выполняют эту задачу с помощью извлечения статистических корреляций высокого порядка в ограничивающих векторах. Сеть, описанная выше, является частным случаем машины Больцмана. Ее можно рассматривать как процедуру обучения без учителя моделированию распределения вероятности, которое применяется к видимым нейронам с соответствующими вероятностями. Таким образом, сеть может осуществлять *дополнение образов* (pattern completion). В частности, если вектор с неполной информацией поступает в подмножество видимых нейронов, сеть (в предположении правильности процедуры обучения) дополняет эту информацию в оставшихся видимых нейронах [457].

Главной целью обучения Больцмана является создание нейронной сети, которая правильно моделирует входные образы в соответствии с распределением Больцмана. При использовании этой формы обучения делаются два предположения.

- Каждый входной вектор внешней среды подается на вход сети достаточно долго, чтобы система достигла *температурного равновесия*.
- Не существует определенной последовательности подачи векторов среды в видимые элементы сети.

---

[6] Видимые нейроны в машине Больцмана можно дополнительно подразделить на входные и выходные. В этой второй конфнгурацин машина Больцмана реализует *ассоциации* под руководством учнтеля. Входные нейроны получают информацию от окружающей среды, а выходные доводят результаты вычисленнй до конечного пользователя.

Считается, что множество синаптических весов реализует совершенную модель структуры среды, если она приводит к точно такому же распределению вероятности состояний видимых элементов (при свободной работе сети), к какому приводит подача входных векторов среды. В общем случае, если количество скрытых нейронов не является экспоненциально большим по сравнению с количеством видимых элементов, такой совершенной модели достичь невозможно. Если же среда имеет упорядоченную структуру, а сеть использует скрытые элементы для извлечения этих закономерностей, можно достичь хорошего соответствия при достаточном количестве скрытых элементов.

Квантование Гиббса и моделирование отжига в машине Больцмана

Обозначим символом $\mathbf{x}$, состоящим из компонентов x_i, соответствующих состояниям нейронов i, вектор состояний машины Больцмана. Это состояние $\mathbf{x}$ представляет собой реализацию случайного вектора $\mathbf{X}$. Синаптические связи между нейронами i и j обозначим символами w_{ji}. При этом

$$w_{ji} = w_{ij} \text{ для всех пар } (i, j) \tag{11.37}$$

и

$$w_{ii} = 0 \text{ для всех } i. \tag{11.38}$$

Равенство (11.37) описывает свойство симметрии, а равенство (11.38) — отсутствие собственных обратных связей. Использование *порога* (bias) достигается за счет добавления веса связи w_{j0} между фиктивным узлом с постоянным сигналом $+1$ и нейроном j (для всех j).

По аналогии с термодинамикой энергия машины Больцмана определяется следующим образом[7]:

$$E(\mathbf{x}) = -\frac{1}{2}\sum_{i}\sum_{j,i\neq j} w_{ji}x_i x_j. \tag{11.39}$$

Вводя распределение Гиббса (11.5), можно определить вероятность того, что сеть находится в состоянии $\mathbf{x}$ (при этом предполагается, что она находится в равновесии

[7] Формула (11.39) применима к машине Больцмана, в которой состояния "вкл" и "выкл" задаются значениями $+1$ и -1 соответственно. Если машина использует для этой цели значения 0 и 1, получим следующее:

$$E(\mathbf{x}) = -\sum_{i}\sum_{j,i\neq j} w_{ji}x_i x_j.$$

при температуре T):

$$P(\mathbf{X}=\mathbf{x})=\frac{1}{Z}\exp\left(-\frac{E(\mathbf{x})}{T}\right), \tag{11.40}$$

где Z — *функция разбиения* (partition function).

Для упрощения выкладок определим событие A и связанные события B и C следующим образом:

A: $X_j=x_j$,

B: $\{X_i=x_i\}_{i=1,i\neq j}^{K}$,

C: $\{X_i=x_i\}_{i=1}^{K}$.

В результате совместное событие B исключает событие A, а совместное событие C включает оба события — A и B. Вероятность B является граничной вероятностью C по отношению к A. Исходя из этого, используя выражения (11.39) и (11.40), можно записать:

$$P(C)=(A,B)=\frac{1}{Z}\exp\left(\frac{1}{2T}\sum_{i}\sum_{j,i\neq j}w_{ji}x_ix_j\right) \tag{11.41}$$

и

$$P(B)=\sum_{A}(A,B)=\frac{1}{Z}\sum_{x_j}\exp\left(\frac{1}{2T}\sum_{i}\sum_{j,i\neq j}w_{ji}x_ix_j\right). \tag{11.42}$$

Экспоненты в равенствах (11.41) и (11.42) могут быть выражены суммами двух компонентов — первая из них включает x_j, а вторая не зависит от x_j. Компонент, содержащий x_j, имеет следующий вид:

$$\frac{x_j}{2T}\sum_{i,i\neq j}w_{ji}x_i.$$

Соответственно, принимая $x_j=x=\pm 1$, можно выразить условную вероятность A для данного B следующим образом:

$$P(A|B)=\frac{P(A,B)}{P(B)}=\frac{1}{1+\exp\left(-\frac{x_j}{T}\sum\limits_{i,i\neq j}w_{ji}x_i\right)}.$$

Это значит, что можно записать:

$$P(X_j=x|\{X_i=x_i\}_{i=1,i\neq j}^{K}=\varphi\left(\frac{x}{T}\sum_{i,i\neq j}w_{ji}x_i\right), \tag{11.43}$$

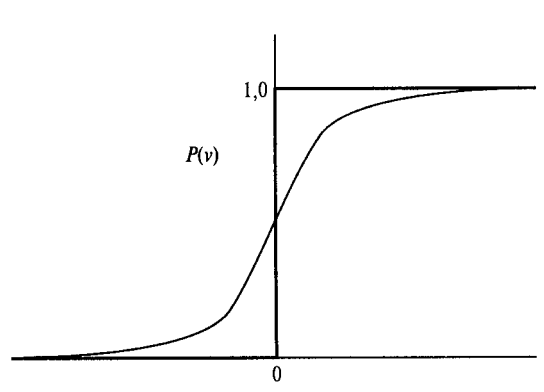


Рис. 11.5. Сигмоидальная функция $P(v)$

где $\varphi(\cdot)$ — сигмоидальная функция своего аргумента, т.е.

$$\varphi(v) = \frac{1}{1 + \exp(-v)}. \tag{11.44}$$

Обратите внимание на то, что, несмотря на изменение x в диапазоне между значениями -1 и $+1$, весь аргумент $\frac{x}{T}\sum_{i \neq j} w_{ji}x_i$ при больших N может варьироваться между $-\infty$ и $+\infty$ (рис. 11.5).

Заметим также, что при выводе равенства (11.43) не использовалась функция разбиения Z. Этот результат очень желателен, так как прямое вычисление Z невозможно в сетях большой сложности.

Использование квантования Гиббса обеспечивает совместное распределение $P(A, B)$. В своей основе (см. раздел 11.6) стохастическое моделирование начинается с присвоения сети некоторого произвольного состояния, после чего нейроны посещаются в естественном порядке. При каждом посещении нейрона выбирается новое значение состояния, в соответствии с распределением вероятности этого нейрона, в зависимости от состояний всех остальных нейронов сети. Предполагая, что стохастическое моделирование будет проводиться достаточно долго, сеть достигнет термального равновесия при температуре T.

К сожалению, время, затраченное на достижение термального равновесия, может быть слишком большим. Для того чтобы избежать этой пролблемы, используется моделирование отжига для конечной последовательности температур $T_0, T_1, \ldots, T_{\text{конечная}}$ (см. раздел 11.5). Таким образом, начальная температура устанавливается в значение T_0, обеспечивая быстрое достижение термального равновесия. После этого температура T медленно снижается до своего окончательного значения $T_{\text{конечная}}$, и в этой точке состояния нейронов достигнут (надеемся) своих желаемых граничных распределений.

Правило обучения Больцмана

Так как машина Больцмана является стохастической, для поиска индекса ее производительности следует обратиться к теории вероятности. Один из таких критериев носит название *функции правдоподобия*[8] (likelihood function). Используя этот критерий в качестве основы, в соответствии с *принципом максимального правдоподобия*, можно определить цель обучения Больцмана как максимизацию функции правдоподобия или (эквивалентно) функции логарифмического правдоподобия.

Обозначим символом $\mathbf{T}$ множество примеров обучения, отобранных из интересующего нас распределения. Предполагается, что эти примеры могут иметь два значения. Количество повторений примеров обучения соответствует частоте появления аналогичных случаев на практике. Обозначим символом $\mathbf{x}_\alpha$ подмножество вектора состояний $\mathbf{x}$, соответствующих видимым нейронам сети. Оставшуюся часть вектора $\mathbf{x}$ обозначим символом $\mathbf{x}_\beta$. Она будет соответствовать состояниям скрытых нейронов. Векторы состояний $\mathbf{x}$, $\mathbf{x}_\alpha$ и $\mathbf{x}_\beta$ являются реализациями случайных векторов $\mathbf{X}$, $\mathbf{X}_\alpha$ и $\mathbf{X}_\beta$ соответственно. Существуют две фазы работы машины Больцмана.

- *Положительная фаза.* В этой фазе сеть работает в своем *фиксированном* (clamped) состоянии (т.е. под непосредственным воздействием множества примеров $\mathbf{T}$).
- *Отрицательная фаза.* В этой фазе сеть работает в свободном режиме и не подвержена влиянию среды.

Если предположить, что вектор $\mathbf{w}$ содержит все синаптические веса сети, то вероятность нахождения видимого нейрона в состоянии $\mathbf{x}_\alpha$ равна $P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$. Предполагая, что большинство возможных примеров, содержащихся в множестве обучения $\mathbf{T}$, являются статистически независимыми, общее распределение вероятности можно представить факториальным распределением $\prod_{\mathbf{x}_\alpha \in \mathbf{T}} P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$. Для того чтобы сформулировать функцию логарифмического правдоподобия $L(\mathbf{w})$, возьмем логарифм

---

[8] Традиционно в качестве индекса производительности машины Больцмана использовалась относительная энтропия (или расстояние Кулбека–Лейблера) [9], [464]. Этот критерий реализовывал меру несоответствия между средой и внутренней моделью сети. Он определялся в следующем виде:

$$D_{p^+||p^-} = \sum_\alpha p_\alpha^+ \log\left(\frac{p_\alpha^+}{p_\alpha^-}\right),$$

где p_α^+ — вероятность того, что видимый нейрон находится в состоянии α в момент нахождения сети в фиксированном режиме, а p_α^- — вероятность того, что тот же нейрон находится в состоянии α в момент нахождения сети в свободном режиме. Синаптические веса сети корректируются с целью минимизации величины $D_{p_\alpha^+||p_\alpha^-}$ (см. задачу 11.10).

Принцип минимума дивергенции Кулбека–Лейблера и максимального правдоподобия эквивалентны, когда применяются к множеству примеров обучения. Для того чтобы заметить эту эквивалентность, обратите внимание на то, что дивергенция Кулбека–Лейблера между распределениями f и g определяется по формуле $D_{f||g} = -H(f) - \sum f \log(g)$.

Если распределение f определяется множеством примеров обучения, а модель g дана для оптимизации, то первое слагаемое является константой, а второе — логарифмическим правдоподобием, взятым с обратным знаком, что и доказывает эквивалентность принципов минимума дивергенции Кулбека–Лейблера и максимума правдоподобия.

этого факториального распределения. При этом будем рассматривать $\mathbf{w}$ как вектор неизвестных параметров:

$$L(\mathbf{w}) = \log \prod_{\mathbf{x}_\alpha \in \mathbf{T}} P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) = \sum_{\mathbf{x}_\alpha \in \mathbf{T}} \log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha). \tag{11.45}$$

Для того чтобы определить выражение для граничной вероятности $P(\mathbf{X}_{\alpha=}\mathbf{x}_\alpha)$ в терминах функции энергии $E(\mathbf{x})$, воспользуемся следующими фактами.

- Исходя из (11.40), вероятность $P(\mathbf{X}{=}\mathbf{x})$ равна $\frac{1}{Z}\exp(-E(\mathbf{x})/T)$.
- По определению вектор состояния $\mathbf{x}$ является совместной комбинацией вектора $\mathbf{x}_\alpha$ (содержащего состояния видимых нейронов) и вектора $\mathbf{x}_\beta$ (содержащего состояния скрытых нейронов). Исходя из этого, вероятность нахождения видимых нейронов в состоянии $\mathbf{x}_\alpha$ с любым $\mathbf{x}_\beta$ определяется по следующей формуле:

$$P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) = \frac{1}{Z}\sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right), \tag{11.46}$$

где случайный вектор $\mathbf{X}_\alpha$ является подмножеством $\mathbf{X}$. Функция разбиения Z определяется следующим выражением (см. (11.6)):

$$Z = \sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right). \tag{11.47}$$

Таким образом, подставляя (11.46) и (11.47) в (11.45), получим искомое выражение для функции максимального правдоподобия:

$$L(\mathbf{w}) = \sum_{\mathbf{x}_\alpha \in \mathbf{T}} \left(\log \sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right) - \log \sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right) \right). \tag{11.48}$$

Здесь зависимость от $\mathbf{w}$ содержится в функции энергии $E(\mathbf{x})$ (см. (11.39)).

Дифференцируя $L(\mathbf{w})$ по w_{ji}, в свете (11.39), после перестановки слагаемых получим следующий результат (см. задачу 11.8):

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \frac{1}{T}\sum_{\mathbf{x}_\alpha \in \mathbf{T}} \left(\sum_{\mathbf{x}_\beta} P(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) x_j x_i - \sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) x_j x_i \right). \tag{11.49}$$

Для упрощения выкладок введем два следующих определения:

$$\rho_{ji}^{+} = < x_j x_i >^{+} = \sum_{\mathbf{x}_\alpha \in \mathbf{T}} \sum_{\mathbf{x}_\beta} P(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) x_j x_i \tag{11.50}$$

и

$$\rho_{ji}^{-} = < x_j x_i >^{-} = \sum_{\mathbf{x}_\alpha \in \mathbf{T}} \sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) x_j x_i. \tag{11.51}$$

В некотором смысле первое среднее ρ_{ji}^{+} можно рассматривать как средний *уровень возбуждения* (mean firing rate) или *корреляцию* между состояниями нейронов i и j при работе сети в своей положительной фазе. Аналогично, второе среднее ρ_{ji}^{-} можно рассматривать как *корреляцию* между состояниями нейронов i и j при работе сети в своей отрицательной фазе. Используя эти обозначения, можно упростить выражение (11.49) следующим образом:

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \frac{1}{T} \left(\rho_{ji}^{+} - \rho_{ji}^{-} \right). \tag{11.52}$$

Целью обучения Больцмана является максимизация функции правдоподобия $L(\mathbf{w})$. Для достижения этой цели можно использовать *градиентный спуск* (gradient ascent) и записать:

$$\Delta w_{ji} = \varepsilon \frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \eta(\rho_{ji}^{+} - \rho_{ji}^{-}), \tag{11.53}$$

где η — *параметр скорости обучения* (learning rate parameter); его можно определить в терминах ε и рабочей температуры T следующим образом:

$$\eta = \frac{\varepsilon}{T}. \tag{11.54}$$

Правило градиентного спуска (11.53) называется *правилом обучения Больцмана* (Boltzmann learning rule). Описанное здесь обучение выполняется пакетным методом. Это значит, что изменения в синаптических весах происходят после представления всего множества примеров обучения.

Согласно этому правилу обучения, синаптические веса машины Больцмана корректируются с использованием только локально наблюдаемых значений при следующих двух разных условиях: фиксированной и свободной фазе работы. Это важное свойство обучения Больцмана упрощает архитектуру сети, что особенно чувствуется при работе с большими сетями. Еще одним важным свойством обучения Больцмана (оно может стать приятным сюрпризом) является то, что правило коррекции синап-

тических весов между нейронами i и j не зависит от того, являются ли они оба видимыми, скрытыми или же один является скрытым, а второй — видимым. Все эти прекрасные свойства обучения Больцмана являются результатом глубоких исследований в [464], [465]. В этой работе абстрактная математическая модель машины Больцмана и нейронные сети были связаны с помощью комбинации двух факторов.

- Распределения Гиббса для описания стохастических свойств нейронов.
- Функции энергии, взятой из статистической физики (11.39), используемой для определения распределения Гиббса.

С точки зрения обучения, два слагаемых, составляющих правило обучения Больцмана (11.53), имеют прямо противоположные значения. Первое слагаемое, соответствующее фиксированному состоянию сети, можно рассматривать как *правило обучения Хебба* (Hebbian learning rule). Второе слагаемое, соответствующее свободному состоянию сети, можно рассматривать как слагаемое *забывания* (forgetting term). И в самом деле, правило обучения Больцмана представляет собой не что иное, как *обобщение правила повторяющегося забывания и обучения* (generalization of the repeated forgetting and relearning rule), описанного в [852] для случая симметричных нейронных сетей, не содержащих скрытых нейронов.

Интересно отметить следующее. Так как алгоритм обучения машины Больцмана требует, чтобы скрытые нейроны чувствовали разницу между стимулируемым и свободным режимами, при наличии (скрытой) внешней сети, посылающей сигналы этим скрытым нейронам стимулируемой машины, получим примитивную форму *механизма внимания* (attention mechanism) [227].

Потребность в отрицательной фазе и ее применение

Комбинированное использование положительной и отрицательной фаз стабилизирует распределение синаптических весов в машине Больцмана. Эта потребность может быть обоснована и с другой точки зрения. Интуитивно можно утверждать, что потребность в отрицательной фазе наряду с положительной в машине Больцмана возникает в связи с наличием *функции разделения* (partition function) Z в выражении для вектора вероятности состояний нейрона. Применение этого утверждения заключается в том, что направление наискорейшего спуска в пространстве энергии *отличается* от направления наискорейшего спуска в пространстве параметров. В результате для учета этого несоответствия и возникла потребность в отрицательной фазе [778].

Использование отрицательной фазы в обучении Больцмана имеет два существенных недостатка.

1. *Увеличенное время вычислений.* Во время положительной фазы отдельные нейроны фиксируются внешней средой, в то время как во время отрицательной фазы

все нейроны работают свободно. В результате этого время, затраченное стохастическим моделированием машины Больцмана, возрастает.

2. *Чувствительность к статистическим ошибкам.* Правило обучения Больцмана учитывает различия между двумя средними корреляциями. Одна из них вычисляется для положительной фазы, а вторая — для отрицательной. Когда эти две корреляции подобны, наличие шума делает это различие еще более заметным.

Эти недостатки машины Больцмана можно обойти с помощью использования *сигмоидальных сетей доверия* (sigmoid belief network). В этом новом классе стохастических машин управление процедурой обучения осуществляется посредством фазы, отличной от отрицательной.

11.8. Сигмоидальные сети доверия

Сигмоидальные сети доверия (sigmoid belief network) или *логистические сети доверия* (logistic belief net) были разработаны в [778] в попытке найти стохастическую машину, которая была бы способна обучаться произвольному распределению вероятности на множестве двоичных векторов, но не имела бы, в отличие от машины Больцмана, потребности в отрицательной фазе. Эта цель была достигнута заменой симметричных связей в машине Больцмана *прямыми соединениями, формирующими ацикличный граф.* Говоря более точно, сигмоидальные сети доверия имеют многослойную архитектуру, состоящую из двоичных стохастических нейронов (рис. 11.6). Ацикличная природа этих машин облегчает осуществление вероятностных вычислений. В частности, эта сеть использует сигмоидальную функцию (11.43), по аналогии с машиной Больцмана, для вычисления условной вероятности того, что нейрон будет активирован в ответ на свое собственное индуцированное локальное поле.

Фундаментальные свойства сигмоидальных сетей доверия

Пусть вектор $\mathbf{X}$, состоящий из случайных двоичных переменных $X_1, X_2, \ldots, X_N$, определяет сигмоидальную сеть доверия, состоящую из N стохастических нейронов. *Родители* элемента X_j в векторе $\mathbf{X}$ обозначаются следующим образом:

$$\text{pa}(X_j) \subseteq \{X_1, X_2, \ldots, X_{j-1}\}. \tag{11.55}$$

Другими словами, значением $\text{pa}(X_j)$ является наименьшее подмножество случайного вектора $\mathbf{X}$, для которого

$$P(X_i = x_j | X_1 = x_1, \ldots, X_{j-1} = x_{j-1}) = P(X_j = x_j | \text{pa}(X_j)). \tag{11.56}$$

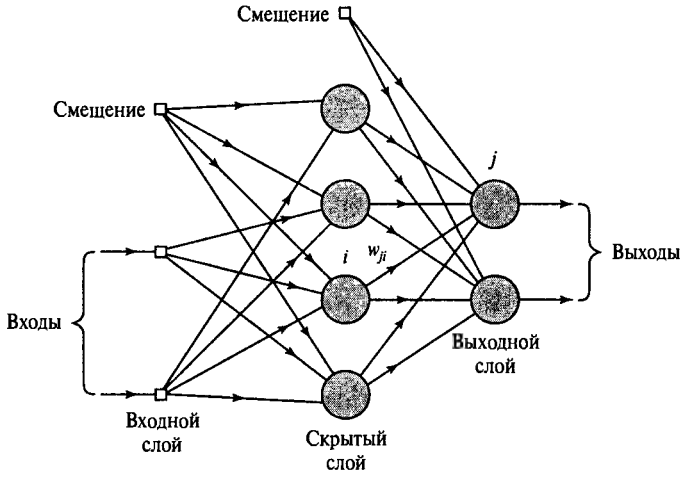


Рис. 11.6. Архитектурный граф сигмоидальной сети доверия

Важным достоинством сигмоидальных сетей доверия является их способность явно представлять условные зависимости исследуемых вероятностных моделей входных данных. В частности, вероятность активации i-го нейрона определяется следующей сигмоидальной функцией (см. (11.43)):

$$P(X_j = x_j|\text{pa}(X_j)) = \varphi\left(\frac{x_j}{T}\sum_{i<j} w_{ji}x_i\right), \tag{11.57}$$

где w_{ji} — синаптический вес, идущий от нейрона i к нейрону j (см. рис. 11.6). Это значит, что условная вероятность $P(X_j = x_j|\text{pa}(X_j))$ зависит от $\text{pa}(X_j)$ только через сумму взвешенных входов. Таким образом, выражение (11.57) создает базис для распространения доверия по сети.

При осуществлении расчетов вероятности в сигмоидальных сетях доверия нужно учесть два следующих условия.

1. $w_{ji} = 0$ для всех X_i, не принадлежащих $\text{pa}(X_j)$.
2. $w_{ji} = 0$ для всех $i \geq j$.

Первое условие вытекает непосредственно из определения родителей. Второе условие следует из того факта, что сигмоидальные сети доверия являются направленным ациклическим графом.

Как следует из самого названия, сигмоидальные сети доверия принадлежат к общему классу *сетей доверия* (belief network)[9], интенсивно изучаемых в литературе [822]. Стохастическая работа сигмоидальных сетей доверия немного сложнее работы машин Больцмана. Тем не менее обе эти системы используют обучение методом градиентного спуска в пространстве вероятностей, основанное на локально доступной информации.

Обучение в сигмоидальных сетях доверия

Обозначим символом $\mathbf{T}$ множество примеров обучения, отобранных из интересующего нас распределения вероятности. Предполагается, что каждый из примеров является двоичным и представляет некоторый атрибут. При обучении допускаются повторения примеров с частотой, пропорциональной частоте встречи на практике подобной комбинации атрибутов. Для моделирования распределения, из которого отобрано множество $\mathbf{T}$, выполним следующие действия.

1. Для сети определим размер вектора состояний $\mathbf{x}$.
2. Выберем подмножество этого вектора (обозначенное $\mathbf{x}_\alpha$), представляющее атрибуты примеров обучения. Это значит, что $\mathbf{x}_\alpha$ представляет собой вектор состояний видимых нейронов.
3. Оставшаяся часть вектора состояний (обозначенная $\mathbf{x}_\beta$) определяет вектор состояний скрытых нейронов (т.е. расчетных узлов, для которых значения не устанавливаются).

Архитектура сигмоидальной сети доверия в значительной мере зависит от организации состояний видимых и скрытых элементов в векторе $\mathbf{x}$. Таким образом, разные композиции состояний скрытых и видимых нейронов могут привести к разным конфигурациям.

Как и в случае с машиной Больцмана, мы выведем искомое правило обучения для сигмоидальной сети доверия, максимизировав функцию логарифмического правдоподобия, вычисленную на множестве примеров $\mathbf{T}$. Для удобства представления еще раз приведем функцию логарифмического правдоподобия $L(\mathbf{w})$ (см. (11.45)):

$$L(\mathbf{w}) = \sum_{\mathbf{x}_\alpha \in \mathbf{T}} \log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha),$$

[9] Сети доверия изначально были введены с целью представления вероятностных знаний в экспертных системах [822]. В литературе они также иногда называются *сетями Байеса, или байесовскими сетями* (Bayesian network).

где $\mathbf{w}$ — вектор синаптических весов сети, который считается неизвестным. Вектор состояний $\mathbf{x}_\alpha$, относящийся к видимым нейронам, является реализацией случайного вектора $\mathbf{X}_\alpha$. Обозначим символом w_{ji} ji-й элемент вектора $\mathbf{w}$ (т.е. синаптический вес, направленный от нейрона i к нейрону j). Дифференцируя функцию $L(\mathbf{w})$ по w_{ji}, получим:

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \sum_{\mathbf{x}_\alpha \in \mathbf{T}} \frac{1}{P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)} \frac{\partial P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)}{\partial w_{ji}}.$$

Далее вспомним два вероятностных соотношения:

$$P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) = \sum_{\mathbf{x}_\beta} P(\mathbf{X} = (\mathbf{x}_\alpha, \mathbf{x}_\beta)) = \sum_{\mathbf{x}_\beta} P(\mathbf{X} = \mathbf{x}), \tag{11.58}$$

где случайный вектор $\mathbf{X}$ относится ко всей сети, а вектор состояний $\mathbf{x}$=($\mathbf{x}_\alpha$, $\mathbf{x}_\beta$) является его реализацией, и

$$P(\mathbf{X} = \mathbf{x}) = P(\mathbf{X} = \mathbf{x} | \mathbf{X}_\alpha = \mathbf{x}_\alpha) P(\mathbf{X}_\alpha = \mathbf{x}_\alpha), \tag{11.59}$$

где определяется вероятность совместного события $\mathbf{X}$=$\mathbf{x}$=($\mathbf{x}_\alpha$, $\mathbf{x}_\beta$).

В свете этих двух соотношений можно определить частную производную $\partial L(\mathbf{w})/\partial w_{ji}$ в эквивалентном виде:

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \sum_{\mathbf{x}_\alpha \in \mathbf{T}} \sum_{\mathbf{x}_\beta} \frac{P(\mathbf{X} = \mathbf{x} | \mathbf{X}_\alpha = \mathbf{x}_\alpha)}{P(\mathbf{X} = \mathbf{x})} \frac{\partial P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)}{\partial w_{ji}}. \tag{11.60}$$

В свете выражения (11.43) можно записать:

$$P(\mathbf{X} = \mathbf{x}) = \prod_j \varphi\left(\frac{x_j}{T} \sum_{i<j} w_{ji} x_i\right), \tag{11.61}$$

где $\varphi(\cdot)$ — сигмоидальная функция своего аргумента. Отсюда следует:

$$\begin{aligned}\frac{1}{P(\mathbf{X} = \mathbf{x})} \frac{\partial P(\mathbf{X} = \mathbf{x})}{\partial w_{ji}} &= \frac{\partial}{\partial w_{ji}} \log P(\mathbf{X} = \mathbf{x}) = \frac{\partial}{\partial w_{ji}} \sum_j \log \varphi\left(\frac{x_j}{T} \sum_{i<j} w_{ji} x_i\right) = \\ &= \frac{1}{T} \sum_j \frac{1}{\varphi\left(\frac{x_j}{T} \sum_{i<j} w_{ji} x_i\right)} \varphi'\left(\frac{x_j}{T} \sum_{i<j} w_{ji} x_i\right) x_j x_i,\end{aligned}$$

где $\varphi'(\cdot)$ — первая производная сигмоидальной функции $\varphi(\cdot)$ от своего аргумента. Однако из определения функции $\varphi(\cdot)$ (см. (11.44)) видно, что

$$\varphi'(v) = \varphi(v)\varphi(-v), \tag{11.62}$$

где $\varphi(-v)$ получается из $\varphi(v)$ заменой v на $-v$. Исходя из этого, можно записать, что

$$\frac{1}{P(\mathbf{X}=\mathbf{x})}\frac{\partial P(\mathbf{X}=\mathbf{x})}{\partial w_{ji}} = \frac{1}{T}\sum_j \varphi\left(-\frac{x_j}{T}\sum_{i<j} w_{ji}x_i\right)x_j x_i. \tag{11.63}$$

Следовательно, подставляя (11.63) в (11.60), получим:

$$\frac{\partial L(\mathbf{w})}{\partial w_{ij}} = \frac{1}{T}\sum_{\mathbf{x}_\alpha\in\mathbf{T}}\sum_{\mathbf{x}_\beta} P(\mathbf{X}=\mathbf{x}|\mathbf{X}_\alpha=\mathbf{x}_\alpha)\varphi\left(-\frac{x_j}{T}\sum_{i<j} w_{ji}x_i\right)x_j x_i. \tag{11.64}$$

Для упрощения определим *среднее по множеству* (ensemble average) как

$$\begin{aligned}\rho_{ji} &= <\varphi\left(-x_j\sum_{i<j} w_{ji}x_i\right)x_j x_i> = \\ &= \sum_{\mathbf{x}_\alpha\in\mathbf{T}}\sum_{\mathbf{x}_\beta} P(\mathbf{X}=\mathbf{x}|\mathbf{X}_\alpha=\mathbf{x}_\alpha)\varphi\left(-\frac{x_j}{T}\sum_{i<j} w_{ji}x_i\right)x_j x_i.\end{aligned} \tag{11.65}$$

Это выражение представляет *усредненную корреляцию* (average correlation) между состояниями нейронов i и j, взвешенную множителем $\varphi\left(-\frac{x_j}{T}\sum_{i<j} w_{ji}x_i\right)$. Это среднее значение берется по всем возможным значениям $\mathbf{x}_\alpha$ (выбранным из множества $\mathbf{T}$), равно как и по всем возможным значениям $\mathbf{x}_\beta$. Здесь $\mathbf{x}_\alpha$ относится к видимым нейронам, а $\mathbf{x}_\beta$ — к скрытым.

Градиентный спуск в пространстве параметров осуществляется с помощью определения пошагового изменения синаптических весов w_{ji}:

$$\Delta w_{ji} = \varepsilon\partial L(\mathbf{w})/\partial w_{ji} = \eta\rho_{ji}, \tag{11.66}$$

где $\eta = \varepsilon/T$ — параметр скорости обучения, а ρ_{ji} определяется выражением (11.65). Равенство (11.65) называется *правилом обучения сигмоидальной сети доверия* (learning rule for a sigmoid belief network).

Процедура обучения сигмоидальной сети доверия в сжатом виде представлена в табл. 11.2. В ней обучение осуществляется в пакетном режиме. Это значит, что изменения, применяемые к синаптическим весам сети, проводятся на основе всего множества примеров обучения. В алгоритме, представленном в табл. 11.2, не учтено

ТАБЛИЦА 11.2. Процедура обучения сигмоидальной сети доверия

Инициализация. Сеть инициализируется путем присвоения весам w_{ji} сети случайных значений, равномерно распределенных в диапазоне $[-a, a]$. Обычно значением a является число 0,5.

1. Для данного множества примеров обучения $\mathbf{T}$ видимые нейроны сети фиксируются в состояниях $\mathbf{x}_\alpha$, где $\mathbf{x}_\alpha \in \mathbf{T}$.
2. Для каждого $\mathbf{x}_\alpha$ выполняется отдельное квантование Гиббса при некоторой рабочей температуре T, после чего наблюдается полученный вектор состояний $\mathbf{x}$ всей сети. Предполагая, что моделирование проводится достаточно долго, значения $\mathbf{x}$ для разных классов, содержащихся в $\mathbf{T}$, должны принять условное распределение соответствующего случайного вектора $\mathbf{X}$, соответствующего данному множеству примеров.
3. Вычисляется среднее по множеству:
$$\rho_{ji} = \sum_{\mathbf{x}_\alpha \in \mathbf{T}} \sum_{\mathbf{x}_\beta} P(\mathbf{X} = \mathbf{x} | \mathbf{X}_\alpha = \mathbf{x}_\alpha) x_j x_i \varphi\left(-x_j \sum_{i<j} w_{ji} x_i\right),$$
где случайный вектор $\mathbf{X}_\alpha$ является подмножеством вектора $\mathbf{X}$ и $\mathbf{x}=(\mathbf{x}_\alpha, \mathbf{x}_\beta)$. Векторы $\mathbf{x}_\alpha$ и $\mathbf{x}_\beta$ соответствуют состояниям видимых и скрытых нейронов, x_j является j-м элементом вектора состояний $\mathbf{x}$ (т.е. состоянием нейрона j), а w_{ji} — синаптическим весом, направленным от нейрона i к нейрону j. Сигмоидальная функция $\varphi(\cdot)$ определяется следующим образом:
$$\varphi(v) = \frac{1}{1+\exp(-v)}.$$
4. Каждый из синаптических весов подвергается коррекции на величину
$$\Delta w_{ji} = \eta \rho_{ji},$$
где η — параметр скорости обучения. Эта коррекция должна перемещать синаптические веса сети в направлении градиента в сторону локального максимума функции логарифмического правдоподобия $\mathrm{L}(\mathbf{w})$ в соответствии с принципом максимального правдоподобия.

использование модели отжига. Именно поэтому температура T устанавливается в значение единицы. Тем не менее, как и в машине Больцмана, моделирование отжига в случае необходимости может быть внедрено в процедуру обучения сигмоидальной сети доверия для ускорения достижения точки термального равновесия.

В отличие от машины Больцмана для обучения сигмоидальной сети доверия требуется всего одна фаза. Причиной такого упрощения является то, что нормализация распределений вероятности по векторам состояния выполняется на локальном уровне каждого из нейронов с помощью сигмоидальной функции $\varphi(\cdot)$, а не глобально посредством сложного вычисления функции разбиения Z, при котором учитываются все возможные конфигурации состояний. Как только условное распределение вектора $\mathbf{X}$ для данных значений $\mathbf{x}_\alpha$ из множества примеров обучения $\mathbf{T}$ было корректно

промоделировано с помощью квантования Гиббса, роль отрицательной фазы процедуры обучения машины Больцмана выполняет весовой множитель $\varphi\left(-\frac{x_j}{T}\sum_{i<j} w_{ji}x_i\right)$, участвующий в вычислении усредненной по множеству корреляции ρ_{ji} между нейронами i и j. Когда достигается локальный минимум функции логарифмического правдоподобия, этот весовой множитель становится равным нулю, если сеть обучалась детерминированному отображению; в противном случае его усредняющий эффект сводится к нулю.

В [778] представлены экспериментальные результаты, которые показали, что сигмоидальные сети доверия способны обучаться моделированию нетривиальных распределений. Эти сети способны обучаться быстрее машин Больцмана; такие преимущества сигмоидальных сетей доверия перед машиной Больцмана появились вследствие устранения из процедуры обучения отрицательной фазы.

11.9. Машина Гельмгольца

Сигмоидальные сети доверия реализуют мощную многослойную систему для представления статистических взаимосвязей высокого порядка между сенсорными входами (и обучения без учителя). В *машине Гельмгольца*[10] (Helmholtz machine), впервые описанной в [245] и [460], предлагается другая удачная многослойная среда для достижения аналогичной цели, но уже без использования квантования Гиббса.

Машина Гельмгольца использует два совершенно противоположных множества синаптических связей (рис. 11.7) для случая, когда двухслойная сеть состоит из двоичных стохастических нейронов. Прямые связи, показанные на рисунке сплошными линиями, составляют *модель распознавания* (recognition model). Целью этой модели является производство логических (infer) о распределении вероятности, исходя из всех представленных примеров входных векторов. Обратные связи, показанные на рисунке пунктирными линиями, составляют *порождающую модель* (generative model).

---

[10] Машина Гельмгольца принадлежит к классу нейронных сетей, характеризуемых прямыми и обратными проекциями (forward-backward projection). Идея таких проекций была предложена в [392], где изучалась теория адаптивного резонанса (adaptive resonance theory) [175]. В этой модели прямая адаптивная фильтрация комбинировалась с обратным сопоставлением моделей. При этом происходил адаптивный резонанс (т.е. усиление и пролонгация нейронной активности). В противоположность теории адаптивного резонанса Гроссберга машины Гельмгольца используют статистический подход к самообучающимся системам в качестве одной из составляющих обобщенной модели, нацеленной на точное извлечение структуры входных данных.

Другими тесно связанными с этой задачей работами являются [686], [687]. В первой из них была введена идея свернутых цепей Маркова (folded Markov chain или FMC). За прямыми переходами в них следовали обратные переходы (использующие теорему Байеса) через копию той же цепи. Во второй работе рассматривалась связь между FMC и машиной Гельмгольца.

Среди других работ можно упомянуть [547], в которой прямая модель распознавания и обратная порождающая модель рассматривались аналогично машине Гельмгольца, однако без вероятностного подхода.

В [244] рассматривалось множество различных вариаций машины Гельмгольца, в том числе схемы обучения с учителем.

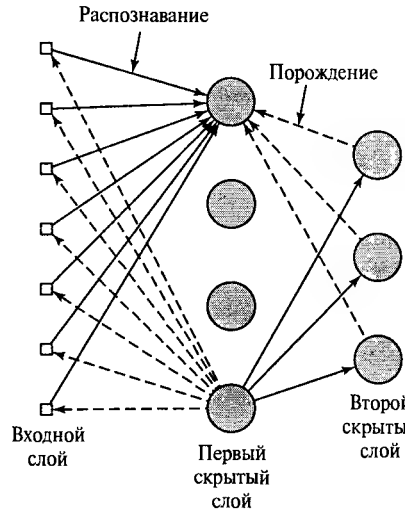


Рис. 11.7. Архитектурный граф машины Гельмгольца, состоящей из взаимосвязанных нейронов со связями распознавания (сплошные линии) и порождающими связями (пунктирные линии)

Эта модель предназначена для аппроксимации исходных входных векторов на основе представлений, созданных скрытыми слоями сети, путем *самоорганизации*. Обе эти модели работают на основе метода прямого распространения, не имеют обратных связей и взаимодействуют друг с другом посредством процедуры обучения.

В [460] описан стохастический алгоритм, названный *алгоритмом засыпания-пробуждения* (wake-sleep algorithm), предназначенный для вычисления связей распознавания и порождающих весов в машине Гельмгольца. Как следует из названия этого алгоритма, он содержит две фазы: пробуждения и засыпания. В фазе пробуждения проход по сети осуществляется в прямом направлении с использованием связей распознавания, вследствие чего в первом скрытом слое создается представление входных векторов. Затем, в свою очередь, следует второе представление первого представления, которое создается во втором скрытом слое, и т.д. для всех скрытых слоев сети. Множество созданных таким способом представлений в различных скрытых слоях сети обеспечивает полное описание множества входных векторов в данной сети. Несмотря на то что сигналы передаются по распознающим весам, на самом деле во время фазы пробуждения на основе локально доступной информации обучаются только порождающие связи. В результате эта фаза процесса обучения повышает качество восстановления для каждого из слоев общего представления, сформированного в предыдущем слое.

Во время фазы засыпания веса распознавания отключаются. Сеть работает послойно в обратном направлении с помощью порождающих весов, начиная с самого дальнего скрытого слоя и заканчивая входным слоем. Благодаря тому что нейроны являются стохастическими, повторение этого процесса обычно приводит к созданию массы "фантастических" векторов во входном слое. Эти "фантазии" составляют

несмещенный образ порождающей модели мира в данной сети. После создания такой "фантазии" для коррекции весов распознавания применяется обычное дельта-правило (см. главу 3). Целью этого является максимизация вероятности восстановления действий скрытых слоев, вызвавших данную фантазию. Подобно фазе пробуждения, фаза засыпания использует только локально доступную информацию.

В процессе обучения порождающих весов (т.е. обратных связей) также используется простое дельта-правило. Однако вместо градиента функции логарифмического правдоподобия в этом правиле используется градиент *штрафной* функции логарифмического правдоподобия (penalized log-likelihood function). Слагаемое штрафа представляет собой дивергенцию Кулбека–Лейблера между истинным апостериорным распределением и фактическим распределением, создаваемым моделью распознавания [460]. Дивергенция Кулбека–Лейблера (или относительная энтропия) рассматривалась в предыдущей главе. В результате штрафная функция логарифмического правдоподобия выступает как нижняя граница функции логарифмического правдоподобия входных данных, и эта нижняя граница улучшается в процессе обучения. В частности, в процессе обучения происходит корректировка порождающих весов с целью получения истинного апостериорного распределения, максимально приближенного к распределению, вычисленному моделью распознавания. К сожалению, обучение весов модели распознавания точно не соответствует штрафной функции правдоподобия. Процедура пробуждения-засыпания не гарантирует работоспособность во всех практических ситуациях — иногда она завершается неудачей.

11.10. Теория среднего поля

Обучаемые машины, рассмотренные в предыдущих трех разделах, имеют одно общее свойство: в них используются стохастические нейроны, вследствие чего можно обеспечить только медленный процесс обучения. В заключительной части настоящей главы рассматривается применение теории среднего поля в качестве математического базиса для вывода *детерминированних приближений* (deterministic approximation) стохастических машин, что позволит значительно ускорить обучение. Так как рассмотренные ранее стохастические машины имеют разную архитектуру, то и теория среднего поля применяется к ним по-разному. В частности, можно определить два особых подхода, которые были предложены в литературе.

1. Корреляция заменяется своей аппроксимацией среднего поля.
2. Нетрактуемая модель заменяется трактуемой с помощью вариационного принципа.

Второй подход — очень принципиальный и потому более привлекательный. Он применим к сигмоидальным сетям доверия [936] и машинам Гельмгольца [245]. Однако в случае машин Гельмгольца применение второго подхода усложняется потреб-

ностью в ограничении сверху функции разбиения Z. По этой причине в [831] для ускорения процесса обучения Больцмана использовался первый подход. В этом разделе будет дано обоснование первому подходу, а вторым подходом мы займемся в следующем разделе.

Идея аппроксимации среднего поля хорошо известна в статистической физике [363]. Хотя нельзя отвергнуть тот факт, что в контексте стохастических машин желательно знать состояния всех нейронов сети в любой момент времени, тем не менее понятно, что в сетях с большим количеством нейронов их состояния содержат значительно бо́льшую информацию, чем это требуется на практике. На самом деле, для того чтобы ответить на наиболее известные вопросы физики о стохастическом поведении сети, требуется знать только средние значения состояний нейронов или средние произведения пар состояний нейронов.

В стохастическом нейроне механизм возбуждения описывается вероятностным правилом. В этом случае целесообразно будет говорить о *среднем значении* состояний x_j нейрона j. Выражаясь более точно, будем говорить о "среднем" как о "термальном среднем", поскольку синаптический шум обычно моделируется в терминах термальных флуктуаций. В любом случае обозначим символом $<x_j>$ среднее значение x_j. Состояние нейрона описывается следующим вероятностным правилом:

$$x_j = \begin{cases} +1, & \text{с вероятностью } P(v_j), \\ -1, & \text{с вероятностью } 1 - P(v_j), \end{cases} \tag{11.67}$$

где

$$P(v_j) = \frac{1}{1 + \exp(-v_j/T)}, \tag{11.68}$$

где T — рабочая температура. Исходя из этого, среднее $<x_j>$ для некоторого *заданного* значения индуцированного локального поля v_j можно записать следующим образом:

$$< x_j >= (+1)P(v_j) + (-1)[1 - P(v_j)] = 2P(v_j) - 1 = \tanh(v_j/2T), \tag{11.69}$$

где $\tanh(v_j/2T)$ — гиперболический тангенс своего аргумента. На рис. 11.8 показаны два графика среднего значения $<x_j>$ относительно индуцированного локального поля v_j. Данная непрерывная кривая приведена для некоторой температуры T, большей нуля. Жирной линией показан график для предельного случая T =0. В последнем случае выражение (11.69) принимает свою предельную форму:

$$< x_j > \rightarrow \operatorname{sgn}(v_j)T \rightarrow 0. \tag{11.70}$$

Это соответствует функции активации нейрона Мак-Каллока–Питца.

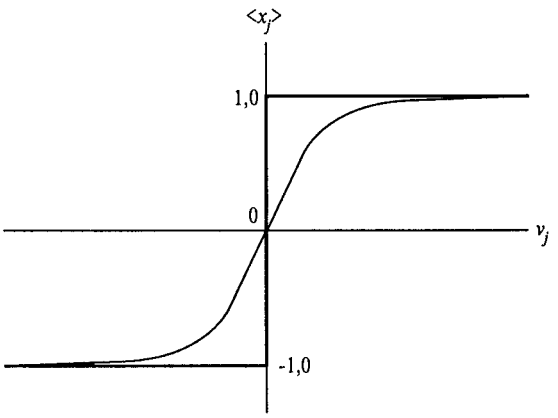


Рис. 11.8. График термального среднего $\langle x_j \rangle$ относительно индуцированного локального поля v_j. Жирная линия соответствует обычному нейрону Мак-Каллока–Питца

До сих пор основное внимание уделялось простейшему случаю обособленного стохастического нейрона. В более общем случае стохастической машины, составленной из множества таких нейронов, приходится сталкиваться с более сложной задачей. Эта сложность проистекает из сочетания двух следующих факторов.

- Вероятность $P(v_j)$ того, что нейрон j включен, является нелинейной функцией индуцированного локального поля v_j.
- Индуцированное локальное поле v_j представляет собой случайную переменную, на которую оказывают стохастическое воздействие другие нейроны, связанные со входом данного.

Можно с уверенностью утверждать, что не существует математического метода, который можно использовать для точной оценки стохастического поведения нейрона. Однако существуют приближения, известные как *аппроксимации среднего поля* (mean-field approximation), использование которых часто приводит к положительным результатам. Основная идея, стоящая за аппроксимацией среднего поля, заключается в замене фактической флуктуации индуцированного локального поля v_j каждого из нейронов сети ее средним значением $<v_j>$:

$$v_j \overset{\text{прибл.}}{=} < v_j >= \left\langle \sum_i w_{ji} x_i \right\rangle = \sum_i w_{ji} < x_i > . \tag{11.71}$$

Следовательно, можно вычислить среднее состояние $<x_j>$ нейрона j, содержащегося в стохастической машине среди N нейронов, так же, как это делалось в выражении (11.69) для обособленного стохастического нейрона:

$$< x_j > = \tanh\left(\frac{1}{2T} v_j\right) \overset{\text{прибл.}}{=} \tanh\left(\frac{1}{2T} < v_j >\right) =$$

$$= \tanh\left(\frac{1}{2T} \sum_i w_{ji} < x_j >\right). \qquad (11.72)$$

В свете выражения (11.72) можно определить аппроксимацию среднего поля следующим образом.

Среднее некоторой функции случайной переменной аппроксимируется функцией среднего значения этой случайной переменной.

Для $j = 1, 2, \ldots, N$ выражение (11.72) представляет множество нелинейных уравнений с N неизвестными $<x_j>$. Решение этой системы нелинейных уравнений является вполне посильной задачей, так как все неизвестные являются *детерминированными*, а не стохастическими переменными, какими они были в исходной сети.

11.11. Детерминированная машина Больцмана

Время обучения машины Больцмана *экспоненциально* зависит от количества нейронов, так как правило обучения Больцмана требует вычисления корреляций между всеми парами нейронов сети. Таким образом, обучение Больцмана требует экспоненциального времени. В [831] был предложен метод ускорения процесса обучения Больцмана, в котором предлагается заменить корреляцию в правиле обучения Больцмана (11.53) следующей аппроксимацией среднего поля:

$$< x_j x_i > \overset{\text{прибл.}}{=} < x_j >< x_i >, \qquad (i, j) = 1, 2, \ldots, K, \qquad (11.73)$$

где само среднее значение $<x_j>$ вычисляется с использованием уравнения среднего поля (11.72).

Эта форма обучения Больцмана, в которой вычисления корреляций аппроксимируются только что описанным способом, получила название *детерминированного правила обучения Больцмана* (deterministic Boltzmann learning rule). В частности, стандартное правило обучения Больцмана (11.53) аппроксимируется следующим образом:

$$\Delta w_{ji} = \eta(U_j^+ U_i^+ - U_j^- U_i^-), \qquad (11.74)$$

где U_j^+ и U_j^- — усредненные выходы видимых нейронов j в фиксированном и свободном режимах соответственно, а η — параметр скорости обучения. В то время как машина Больцмана использует двоичные стохастичные нейроны, ее детерминированный "родственник" использует аналогичные детерминированные нейроны.

Детерминированная машина Больцмана показала повышение скорости по сравнению со стандартной машиной Больцмана на один-два порядка [831]. Однако при ее практическом использовании следует учесть следующее.

1. Детерминированное правило обучения Больцмана работает только с учителем. Это значит, что отдельным видимым нейронам присваивается роль выходных нейронов. Обучение без учителя не работает ни в одном режиме среднего поля, так как среднее значение состояний является очень маломощным представлением свободного распределения вероятности.
2. При обучении с учителем использование детерминированного правила обучения Больцмана ограничено сетями только с одним скрытым слоем [332]. Теоретически отсутствуют основания не помещать в такие сети несколько скрытых слоев. Однако на практике использование нескольких скрытых слоев приводит к той же проблеме, которая для обучения без учителя была упомянута в п. 1.

Детерминированное правило обучения Больцмана (11.74) имеет простую и локальную форму, которая представляет его отличным кандидатом на внедрение в *VLSI-устройства* (very large scale integration) [19], [942].

11.12. Детерминированные сигмоидальные сети доверия

Сущность метода аппроксимации среднего поля, описанного в разделе 11.10, состоит в том, что среднее некоторой функции случайной переменной аппроксимируется этой же функцией от среднего значения случайной переменной. Эта точка зрения теории среднего поля имеет ограниченное применение для аппроксимации машины Больцмана, что уже было показано в предыдущем разделе. В настоящем разделе мы рассмотрим другую точку зрения теории среднего поля, которая лучше всего подходит для аппроксимации сигмоидальных сетей доверия. Покажем, что нетрактуемая модель аппроксимируется трактуемой с помощью вариационного принципа [524], [936]. Грубо говоря, трактуемая модель характеризуется изменением числа степеней свободы, что делает исходную модель нетрактуемой. Это нарушение связей происходит за счет расширения нетрактуемой модели для включения дополнительных параметров, известных под названием *вариационных* (variational parameters), которые предназначены для адаптации решаемой задачи. Терминология, используемая в этом подходе, уходит своими корнями в *вариационное исчисление* (calculus of variations) [814].

Нижняя граница функции логарифмического правдоподобия

В начале обсуждения напомним вероятностное соотношение (11.58), которое воспроизведем здесь в логарифмическом виде:

$$\log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) = \log \sum_{\mathbf{x}_\beta} P(\mathbf{X} = \mathbf{x}). \tag{11.75}$$

Как и в разделе 11.8, разобьем случайный вектор $\mathbf{X}$ на $\mathbf{X}_\alpha$ и $\mathbf{X}_\beta$, соответствующие видимым и скрытым нейронам. Реализации этих случайных векторов обозначим $\mathbf{x}$, $\mathbf{x}_\alpha$ и $\mathbf{x}_\beta$. Заметим, что с логарифмом суммы вероятностей (11.75) сложно работать. Эту сложность можно обойти, заметив, что для любого условного распределения $Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)$ равенство (11.75) можно переписать в новом эквивалентном виде:

$$\log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) = \log \sum_{\mathbf{x}_\beta} Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) \left[\frac{P(\mathbf{X} = \mathbf{x})}{Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)} \right]. \tag{11.76}$$

Это равенство сформулировано для применения *неравенства Йенсена* (Jensen's inequality), о котором уже говорилось в предыдущей главе. Используя его для данного случая, получим:

$$\log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha) \geq \sum_{\mathbf{x}_\beta} Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) \log \left[\frac{P(\mathbf{X} = \mathbf{x})}{Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)} \right]. \tag{11.77}$$

Исходя из терминологии теории среднего поля, назовем приближенное распределение $Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)$ *распределением среднего поля* (mean-field distribution).

В данном случае нас интересует формула функции логарифмического правдоподобия. В сигмоидальных сетях доверия функция логарифмического правдоподобия $L(\mathbf{w})$ определяется суммированием по всем $\mathbf{x}_\alpha$ (определенным на множестве примеров $\mathbf{T}$). Поэтому для обучения сети применяется пакетный алгоритм. Для аппроксимации среднего поля сигмоидальной сети доверия будем использовать другую стратегию. Попытаемся адаптировать последовательный режим, в котором функция логарифмического правдоподобия вычисляется после *подачи в сеть каждого примера*:

$$\mathbf{L}(\mathbf{w}) = \log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha), \tag{11.78}$$

где $\mathbf{w}$ — вектор весов сети. В случае идентично и независимо распределенных данных реальная функция логарифмического правдоподобия $\mathbf{L}(\mathbf{w})$ представляет собой сумму слагаемых $\mathbf{L}(\mathbf{w})$ по одному для каждой точки данных. В этой ситуации определения $\mathbf{L}(\mathbf{w})$ и $L(\mathbf{w})$ эквивалентны. В общем же случае использование $\mathbf{L}(\mathbf{w})$ предоставляет аппроксимацию функции $L(\mathbf{w})$.

Последовательный подход (реального времени) к обучению стал стандартом для нейронных сетей в основном благодаря простоте своей реализации. В свете (11.78) можно записать:

$$\mathbf{L}(\mathbf{w}) \geq \sum_{\mathbf{x}_\beta} Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) \log \left[\frac{P(\mathbf{X} = \mathbf{x})}{Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)} \right],$$

или, эквивалентно:

$$\begin{aligned} \mathbf{L}(\mathbf{w}) \geq & - \sum_{\mathbf{x}_\beta} Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) \log Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) \\ & + \sum_{\mathbf{x}_\beta} Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) \log P(\mathbf{X} = \mathbf{x}). \end{aligned} \tag{11.79}$$

Первое слагаемое в правой части (11.79) представляет собой энтропию распределения среднего поля $Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)$. Не следует путать его с условной энтропией! Второе слагаемое представляет собой среднее логарифма $\log P(\mathbf{X}=\mathbf{x})$ по всем возможным состояниям скрытых нейронов. При единичной температуре (см. рассуждения о распределении Гиббса в разделе 11.2) энергия сигмоидальной сети доверия равна $-log\ P(\mathbf{X} = \mathbf{x})$. Таким образом, с учетом (11.61) при $T = 1$, получим:

$$P(\mathbf{X} = \mathbf{x}) = \prod_j \varphi \left(x_j \sum_{i<j} w_{ji} x_i \right).$$

Следовательно,

$$E = -\log P(\mathbf{X} = \mathbf{x}) = -\sum_j \log \varphi \left(x_j \sum_{i<j} w_{ji} x_i \right). \tag{11.80}$$

Используя определение сигмоидальной функции

$$\varphi(v) = \frac{1}{1 + \exp(-v)} = \frac{\exp(v)}{1 + \exp(v)},$$

функцию энергии сигмоидальной сети доверия можно формально выразить следующим образом:

$$E = -\sum_i \sum_{j,j<i} w_{ji} x_i x_j + \sum_j \log \left(1 + x_j \sum_{i<j} w_{ji} x_i \right). \tag{11.81}$$

За исключением множителя 1/2, в первом слагаемом правой части (11.81) можно узнать функцию энергии Марковской системы (например, машины Больцмана). Однако второе слагаемое уникально только для сигмоидальных сетей доверия.

Нижняя граница (11.79) подходит для любого распределения среднего поля $Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)$. Однако для удобства использования необходимо выбрать такое распределение, которое позволит оценить эту границу. Для этого используем *факториальное распределение* (factorial distribution) [936]:

$$Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha) = \prod_{j \in \mathbf{H}} \mu_j^{x_j} (1 - \mu_j)^{1 - x_j}, \tag{11.82}$$

где $\mathbf{H}$ — множество всех скрытых нейронов, а состояния скрытых нейронов представляют собой независимые *переменные Бернулли* (Bernoulli variable) с корректируемым средним μ_j. (Bernoulli (θ) определяется как двоичная случайная переменная, принимающая значение 1 с вероятностью θ.) Исходя из этого, подставляя (11.82) в (11.79), после упрощения получим:

$$\begin{aligned} \mathbf{L}(\mathbf{w}) \geq & - \sum_{j \in \mathbf{H}} [\mu_j \log \mu_j + (1 - \mu_j) \log(1 - \mu_j)] + \\ & + \sum_i \sum_{j \in \mathbf{H}, i<j} w_{ji} \mu_i \mu_j - \sum_{j \in \mathbf{H}} \left\langle \log \left[1 + \exp \left(\sum_{i<j} w_{ji} x_i \right) \right] \right\rangle, \end{aligned} \tag{11.83}$$

где под <·> подразумевается среднее по множеству распределение среднего поля, а выражение $j \in \mathbf{H}$ означает принадлежность данного нейрона к скрытому слою. Первое слагаемое в правой части (11.83) является энтропией среднего поля, а второе — энергией среднего поля. Оба эти слагаемые относятся к факториальному распределению (11.82).

К сожалению, остается одна нерешенная проблема: невозможно вычислить точное среднее формы $\langle \log[1 + \exp(z_j)] \rangle$. Это слагаемое содержится в (11.83) с учетом подстановки

$$z_j = \sum_{i<j} w_{ji} x_i. \tag{11.84}$$

Чтобы обойти эту сложность, снова обратимся к неравенству Йенсена. Прежде всего, для любой переменной z_j и произвольного действительного числа ξ_j $\langle log[1 + \exp(z_j)] \rangle$ можно выразить в отличной, но эквивалентной форме:

$$\begin{aligned} < \log(1 + e^{z_j}) > &= < \log[e^{\xi_j z_j} e^{-\xi_j z_j} (1 + e^{z_j})] >= \\ &= \xi_j < z_j > + < \log[e^{-\xi_j z_j} + e^{(1-\xi_j) z_j}] >, \end{aligned} \tag{11.85}$$

где $<z_j>$ — среднее по множеству значение z_j. Теперь применим неравенство Йенсена в другой форме. Ограничивая сверху среднее в правой части (11.85), получим:

$$1 < \log(1 + e^{z_j}) >\leq \xi_j < z_j > + \log < e^{-\xi_j z_j} + e^{(1-\xi_j)z_j} > . \qquad (11.86)$$

Присваивая ξ_j значение нуль, получим соотношение

$$< \log(1 + e^{z_j}) >\leq \log < 1 + e^{z_j} > .$$

Применяя ненулевое значение ξ_j, можно ограничить среднее $<\log[1+\exp(z_j)]>$ более точно, чем это возможно при стандартном ограничении [965]. Это будет продемонстрировано на следующем примере.

Пример 11.3

Переменная с гауссовым распределением

Для демонстрации полезности ограничения (11.86) рассмотрим пример переменной с гауссовым распределением, нулевым средним и единичной дисперсией. Для этого частного случая точным значением $<\log[1+\exp(z_j)]>$ является 0,806. Ограничение, описываемое (11.86), принимает вид $e^{0.5\xi^2} + e^{0.5(1-\xi^2)}$. Оно достигает своего минимального значения 0,818 при ξ =0,5. Это значение гораздо ближе к истиному значению, чем значение 0,974, полученное из стандартного ограничения при ξ =0 [936]. ■

Возвращаясь к рассматриваемому вопросу, подставляя (11.85) и (11.86) в (11.83), получим нижнюю границу логарифмического правдоподобия события $\mathbf{X}_\alpha = \mathbf{x}_\alpha$ в следующем виде:

$$\mathbf{L}(\mathbf{w}) \geq -\sum_{j\in\mathbf{H}}[\mu_j \log \mu_j + (1-\mu_j)\log(1-\mu_j)] + \sum_{j\in\mathbf{H}}\sum_{i<j,i<j} w_{ji}\mu_i(\mu_j - \xi_j) - \\ -\sum_{j\in\mathbf{H}} \log\left\langle \exp(-\xi_j z_j) + \exp((1-\xi_j)z_j)\right\rangle, \qquad (11.87)$$

где само z_j определяется выражением (11.84). Это выражение и является искомым ограничением функции логарифмического правдоподобия $\mathbf{L}(\mathbf{w})$, вычисляемой в последовательном режиме алгоритма.

Процедура обучения для аппроксимации среднего поля сигмоидальной сети доверия

При выводе ограничения (11.87) было введено множество *вариационных параметров*: μ_j для $j \in \mathbf{H}$ и ξ_j для всех j без определения их в явном виде. Эти параметры являются настраиваемыми. Так как основной целью является максимизация функции логарифмического правдоподобия, естественно осуществлять поиск таких значений

μ_j и ξ_j, которые максимизируют выражение в правой части (11.87). Для достижения этой цели будем использовать двухшаговую итеративную процедуру, которая описана в [936].

Сначала рассмотрим ситуацию, в которой среднее значение μ_j фиксировано, а требуется найти такие параметры ξ_j, которые обеспечивают самое близкое к реальному ограничение функции логарифмического правдоподобия $\mathbf{L}(\mathbf{w})$. Следует заметить, что выражение в правой части (11.87) *не объединяет* слагаемые с ξ_j, которые относятся к разным нейронам сети. Исходя из этого, минимизация выражения по ξ_j сводится к N независимым операциям минимизации на интервале [0, 1], где N — общее количество нейронов в сети.

Теперь рассмотрим ситуацию, в которой значения ξ_j фиксированы, а требуется найти такое среднее значение μ_j, которое обеспечивает самое точное ограничение функции логарифмического подобия $\mathbf{L}(\mathbf{w})$. С этой целью введем следующее определение:

$$K_{ji} = -\frac{\partial}{\partial \mu_i} \log \left\langle \exp(-\xi_j z_j) + \exp((1-\xi_j) z_j) \right\rangle, \tag{11.88}$$

где случайная переменная z_j определяется выражением (11.84). Частная производная K_{ji} является мерой влияния родительского состояния x_i нейрона i на состояние x_j нейрона j для данного примера $\mathbf{x}_\alpha \in \mathbf{T}$. Как и в случае с синаптическими весами сигмоидальных сетей доверия, K_{ji} будут иметь ненулевое значение только в том случае, когда состояние x_i является родительским по отношению к состоянию x_j. Используя факториальное распределение (11.82), можно оценить среднее по множеству величин $\exp(-\xi_j z_j)$ и $\exp((1-\xi_j) z_j)$, а затем — частную производную K_{ji} (формула для вычисления последнего приведена в табл. 11.3). Имея значение K_{ji}, можно продолжить решение задачи вычисления параметра μ_j, максимизирующего функцию логарифмического правдоподобия $\mathbf{L}(\mathbf{w})$ для фиксированного ξ_j. В частности, дифференцируя (11.87) по μ_j, приравнивая результат к нулю и переставляя слагаемые, получим:

$$\log\left(\frac{\mu_j}{1-\mu_j}\right) = \sum_{i<j} [w_{ji}\mu_i + w_{ij}(\mu_i - \xi_i) + K_{ij}].$$

Эквивалентно, можно записать:

$$\mu_j = \varphi\left(\sum_{i<j} [w_{ji}\mu_i + w_{ij}(\mu_i - \xi_i) + K_{ij}]\right), \quad j \in \mathbf{H}, \tag{11.89}$$

где $\varphi(\cdot)$ — сигмоидальная функция. Равенство (11.89) называется *уравнением среднего поля* (mean-field equation) для сигмоидальной сети доверия. Аргумент сигмоидальной функции в этом уравнении образует так называемое *покрытие Маркова* (Markov blanket), характеризуемое следующим образом.

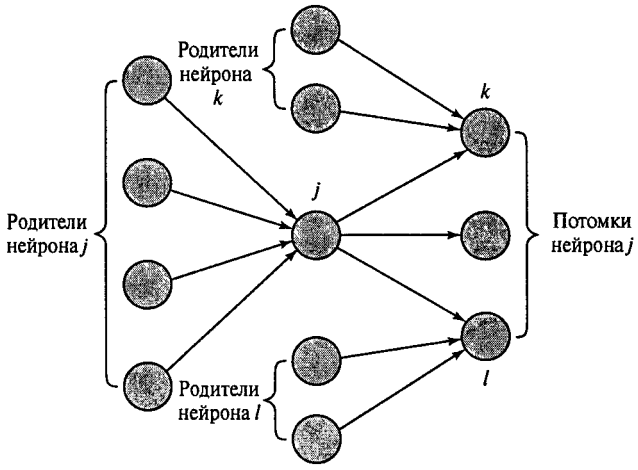


Рис. 11.9. Покрытие Маркова

- Родители и потомки нейрона j представлены слагаемыми $w_{ji}\mu_j$ и $w_{ij}\mu_i$ соответственно.
- Другие родители потомков нейрона j учитываются через частную производную K_{ij}.

Покрытие Маркова нейрона j показано на рис. 11.9. Понятие "покрытия Маркова" было введено в [821]. В этой работе утверждалось, что эффективный вход нейрона j составлен из слагаемых, относящихся к его родителям, потомкам и родителям последних.

Если выполняется условие того, что выбор факториального распределения (11.82) в качестве аппроксимации *апостериорного* распределения $P(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)$ не точен, уравнение среднего поля (11.89) устанавливает параметры $\{\mu_j\}_{j\in H}$ в такие оптимальные значения, которые делают аппроксимацию насколько возможно точной. Это, в свою очередь, приводит к точному ограничению среднего поля функции логарифмического правдоподобия $\mathbf{L}(\mathbf{w})$, вычисляемой в последовательном режиме [936].

После вычисления скорректированных значений параметров $\{\xi_j\}$ и $\{\mu_j\}$ переходим к вычислению коррекции синаптических весов w_{ji} по следующей формуле:

$$\Delta w_{ji} = \eta \frac{\partial B(\mathbf{w})}{\partial w_{ji}}, \tag{11.90}$$

где η — параметр скорости обучения; $B(\mathbf{w})$ — нижняя граница функции логарифмического правдоподобия $\mathbf{L}(\mathbf{w})$, т.е. $B(\mathbf{w})$ — выражение в правой части формулы (11.83). Используя эту формулу, несложно вычислить и частные производные $\partial B(\mathbf{w})/\partial w_{ji}$.

Процесс обучения для приближения среднего поля к сигмоидальной сети доверия представлен в табл. 11.3. В этой таблице содержатся формулы для оценки частных производных K_{ji} и $\partial B(\mathbf{w})/\partial w_{ji}$.

ТАБЛИЦА 11.3. Процедура обучения для приближения среднего поля к сигмоидальной сети доверия

Инициализация. Сеть инициализируется присвоением весам w_{ji} сети случайных значений, равномерно распределенных в интервале $[-a, a]$, где в качестве a обычно выбирают число 0.5

Вычисления. Для примера $\mathbf{x}_\alpha$, выбранного из множества обучения $\mathbf{T}$, выполняем следующие вычисления

1. *Коррекция* $\{\xi_j\}$ *при фиксированных* $\{\mu_j\}$

   Фиксируем средние значения $\{\mu_j\}_{j\in\mathbf{H}}$, относящиеся к факториальному приближению апостериорного распределения $P(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)$, и минимизируем следующую границу функции логарифмического правдоподобия:

$$B(\mathbf{w}) = -\sum_{j\in\mathbf{H}}[\mu_j \log \mu_j + (1-\mu_j)\log(1-\mu_j)] + \sum_i \sum_{j\in\mathbf{H}, i<j} w_{ji}\mu_i\mu_j =$$

$$= -\sum_i \sum_{j\in\mathbf{H}, i<j} w_{ji}\mu_i\xi_j - \sum_{j\in\mathbf{H}} \log\langle \exp(-\xi_j z_j) + \exp((1-\xi_j)z_j)\rangle,$$

   где $z_j = \sum_{i<j} w_{ji}x_i$.

   Минимизация $B(\mathbf{w})$ сводится к N независимым операциям минимизации на интервале [0, 1]

2. *Коррекция* $\{\mu_j\}$ *при фиксированных* $\{\xi_j\}$

   Фиксируя значения параметров $\{\xi_j\}$, повторяем уравнения среднего поля:

$$\mu_j = \varphi\left(\sum_{i<j}[w_{ji}\mu_i + w_{ij}(\mu_i - \xi_i) + K_{ij}]\right),$$

   где

$$K_{ji} = -\frac{\partial}{\partial \mu_i}\log\langle \exp(-\xi_j z_j) + \exp((1-\xi_j)z_j)\rangle =$$

$$= \frac{(1-\theta_j)(1-\exp(-\xi_j w_{ji}))}{1-\mu_i+\mu_i\exp(-\xi_j w_{ji})} + \frac{\theta_j(1-\exp((1-\xi_j)w_{ji}))}{1-\mu_i+\mu_i\exp((1-\xi_j)w_{ji})},$$

$$\theta_j = \frac{\langle \exp((1-\xi_j)z_j)\rangle}{\langle \exp(-\xi_j z_j) + \exp((1-\xi_j)z_j)\rangle}, \quad z_j = \sum_{i<j} w_{ji}x_i.$$

   Функция $\varphi(\cdot)$ является сигмоидальной: $\varphi(v) = \frac{1}{1+\exp(-v)}$

3. *Коррекция синаптических весов*

   Для скорректированных параметров $\{\mu_j\}$ и $\{\xi_j\}$ вычисляем коррекции Δw_{ji} синаптических весов w_{ji}:

$$\Delta w_{ji} = \eta\frac{\partial B(\mathbf{w})}{\partial w_{ji}},$$

Окончание табл. 11.3

где η — параметр скорости обучения,

$$\frac{\partial B(\mathbf{w})}{\partial w_{ji}} = -(\xi_j - \mu_j)\mu_i + \frac{(1-\theta_j)\xi_j\mu_i \exp(-\xi_j w_{ji}))}{1-\mu_i+\mu_i \exp(-\xi_j w_{ji})} - \frac{\theta_j(1-\xi_j)\mu_i(1-\exp((1-\xi_j)w_{ji}))}{1-\mu_i+\mu_i \exp((1-\xi_j)w_{ji})},$$

где θ_j уже определено выше. Теперь корректируем синаптические веса:

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

4. *Цикл по всему множеству примеров* **T**
   Проводим все операции 1–3 для всех примеров, содержащихся в обучающем множестве **T**, таким образом максимизируя их правдоподобие за фиксированное число итераций, или до тех пор, пока не обнаружатся признаки чрезмерного соответствия (overfitting) с использованием, к примеру, перекрестной проверки

11.13. Детерминированный отжиг

Теперь можно перейти к завершающему вопросу настоящей главы, касающемуся детерминированного отжига. В разделе 11.5 рассматривался метод моделирования отжига — прием стохастического ослабления, реализующий мощный метод решения невыпуклых задач оптимизации. Однако в них при выборе расписания отжига следует соблюдать осторожность. В частности, глобальный минимум достигается только в том случае, когда снижение температуры происходит не быстрее логарифмической функции. Это требование делает моделирование отжига непрактичным для применения в большинстве приложений. При моделировании отжига производятся случайные перемещения по поверхности энергии. В противоположность этому в *детерминированном отжиге* (deterministic annealing) в саму функцию стоимости или энергию внедряется некоторая форма случайности, которая затем детерминировано оптимизируется на последовательности понижающихся температур [895], [898]. Не следует путать детерминированный отжиг с *отжигом среднего поля* (последний термин иногда используется для ссылки на детерминированную машину Больцмана).

В последующих разделах описывается идея детерминированного отжига в контексте одной из задач обучения без учителя — *кластеризации*[11].

[11] Детерминированный отжиг успешно применялся во многих задачах обучения.
Векторное квантование [737], [897].
Построение статистического классификатора [734].
Нелинейная регрессия, использующая смесь экспертов [870].
Скрытые модели Маркова в задачах распознавания речи [871].

Кластеризация посредством детерминированного отжига

Кластеризация (clustering) — это деление заданного множества точек данных на подгруппы, каждая из которых, насколько это возможно, гомогенна. Обычно кластеризация представляет собой невыпуклую задачу оптимизации, так как практически все функции искажения, используемые в кластеризации, представляют собой невыпуклые функции входных данных. Более того, график функции искажения относительно данных наполнен локальными минимумами, что делает задачу поиска глобального минимума еще более сложной.

В [895], [896] описана вероятностная среда кластеризации с помощью *рандомизации разбиения* (randomization of partition), или *рандомизации правила кодирования* (randomization of the encoding rule). Главный используемый здесь принцип заключается в том, что каждая точка данных *ассоциирована по вероятности* (associated in probability) с некоторым кластером (подгруппой). Для примера пусть случайный вектор $\mathbf{X}$ обозначает *(входной) вектор источника*, а случайный вектор $\mathbf{Y}$ — наилучший *восстановленный вектор* из соответствующей кодовой книги. Отдельные реализации этих двух векторов обозначим соответственно символами $\mathbf{x}$ и $\mathbf{y}$.

Для кластеризации требуется некоторая *мера искажения* (distortion measure), которую обозначим как $d(\mathbf{x}, \mathbf{y})$. Предполагается, что эта мера $d(\mathbf{x}, \mathbf{y})$ должна удовлетворять двум требованиям: быть выпуклой функцией аргумента $\mathbf{y}$ для всех $\mathbf{x}$ и быть конечной для конечного аргумента. Этим мягким требованиям удовлетворяет, к примеру, квадратичная Евклидова мера искажения:

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||^2. \tag{11.91}$$

Ожидаемое искажение (expected distortion) случайных образов определяется по следующей формуле:

$$\begin{aligned} D &= \sum_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) d(\mathbf{x}, \mathbf{y}) = \\ &= \sum_{\mathbf{x}} P(\mathbf{X} = \mathbf{x}) - \sum_{\mathbf{y}} P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) d(\mathbf{x}, \mathbf{y}), \end{aligned} \tag{11.92}$$

где $P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y})$ — совместная вероятность событий $\mathbf{X} = \mathbf{x}$ и $\mathbf{Y} = \mathbf{y}$. Во второй строке (11.92) использована формула совместной вероятности событий:

$$P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) P(\mathbf{X} = \mathbf{x}). \tag{11.93}$$

Скрытая модель Маркова (hidden Markov model) аналогична цепи Маркова в том, что в обоих случаях переход из одного состояния в другое носит вероятностный характер. Однако оии отличаются друг от друга в фундаментальном смысле. В цепи Маркова выходной результат детерминироваи. В скрытых моделях Маркова выходной результат вероятностный, при этом общий результат может находиться в любом из возможных состояний. Таким образом, имея все состояиия скрытой модели Маркова, получаем распределение вероятности всех выходных символов. Скрытые модели Маркова рассматриваются в [514], [865], [866].

Условная вероятность $P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$ называется *ассоциативной вероятностью* (association probability), т.е. вероятностью ассоциирования кодового вектора **y** со входным вектором **x**.

Ожидаемое искажение D обычно минимизируется по свободным параметрам кластеризуемой модели: восстановленному вектору **y** и ассоциативной вероятности $P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$. Эта форма минимизации дает на выходе "жесткое" решение задачи кластеризации. Здесь под жесткостью подразумевается то, что входной вектор **x** назначается ближайшему кодовому вектору **y**. С другой стороны, при детерминированном отжиге задача оптимизации формулируется иначе: как поиск такого распределения вероятности, которое минимизирует ожидаемое искажение при *заданном уровне случайности* (specified level of randomness). В качестве меры уровня случайности будем использовать *энтропию Шеннона* (см. раздел 10.4):

$$H(\mathbf{X}, \mathbf{Y}) = -\sum_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) \log P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}). \tag{11.94}$$

Тогда условная оптимизация ожидаемого искажения выражается как минимизация Лагранжиана:

$$F = D - TH, \tag{11.95}$$

где T — множитель Лагранжа. Из выражения (11.95) вытекает следующее.

- При больших значениях T энтропия H максимизируется.
- При малых значениях T минимизируется ожидаемое искажение D, что приводит к жесткой (неслучайной) кластеризации.
- При средних значениях T минимизация F приводит к балансу между повышением энтропии H и уменьшением ожидаемого искажения D.

И, что более важно, сравнивая (11.95) с (11.11), можно выявить соответствие между задачей условной оптимизации (кластеризации) и статистической механикой (табл. 11.4). В соответствии с этой аналогией величину T будем называть температурой.

Теперь рассмотрим Лагранжиан F. Обратите внимание на то, что совместная энтропия $H(\mathbf{X}, \mathbf{Y})$ может быть разложена в сумму двух слагаемых:

$$H(\mathbf{X}, \mathbf{Y}) = H(\mathbf{X}) + H(\mathbf{Y}|\mathbf{X}),$$

где $H(\mathbf{X})$ — энтропия источника; $H(\mathbf{Y}|\mathbf{X})$ — условная энтропия вектора восстановления **Y** для данного вектора источника **X**. Энтропия источника не зависит от кла-

ТАБЛИЦА 11.4. Соответствие между условной кластеризацией и статистической физикой

Условная оптимизация кластеризации	*Статистическая физика*
Лагранжиан F	Свободная энергия F
Ожидаемое искажение D	Средняя энергия $<E>$
Энтропия Шеннона H	Энтропия H
Множитель Лагранжа T	Температура T

стеризации. Следовательно, ее можно исключить из определения Лагранжиана F и сфокусировать внимание на условной энтропии

$$H(\mathbf{X},\mathbf{Y}) = -\sum_{\mathbf{x}} P(\mathbf{X}=\mathbf{x}) \sum_{\mathbf{y}} P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x}) \log P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x}). \qquad (11.96)$$

В этом выражении ясно видна роль ассоциативной вероятности $P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x})$. Поэтому, помня о соответствии между задачами условной оптимизации кластеризации и статистической физикой и применяя принцип минимума свободной энергии (см. раздел 11.2), обнаруживаем, что минимизация Лагранжиана F по отношению к ассоциативным вероятностям приводит к распределению Гиббса:

$$P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \exp\left(-\frac{d(\mathbf{x},\mathbf{y})}{T}\right), \qquad (11.97)$$

где Z_x — функция разбиения нашей задачи. Ее можно определить следующим образом:

$$Z_{\mathbf{x}} = \sum_{\mathbf{y}} \exp\left(-\frac{d(\mathbf{x},\mathbf{y})}{T}\right). \qquad (11.98)$$

Когда температура T стремится к бесконечности, ассоциативная вероятность достигает равномерного распределения (см. (11.97)). Использование этого утверждения заключается в том, что при достаточно высоких температурах любой входной вектор равно ассоциирован со всеми кластерами. Такая ассоциация может рассматриваться как "предельно нечеткая". В другом предельном случае, когда температура T достигает нуля, ассоциативная вероятность превращается в дельта-функцию. Следовательно, при очень низких температурах классификация сильно усложняется и с вероятностью 1 каждый входной вектор назначается ближайшему вектору кодирования.

Для того чтобы найти минимальное значение Лагранжиана F, подставим распределение Гиббса (11.97) в (11.92) и (11.96) и полученное выражение используем в формуле Лагранжиана F (11.95). Результат будет иметь следующий вид (см. задачу 11.22):

$$F^* = \min_{p(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x})} F = -T \sum_{\mathbf{x}} P(\mathbf{X}=\mathbf{x}) \log Z_{\mathbf{x}}. \tag{11.99}$$

Для того чтобы минимизировать Лагранжиан по оставшимся свободным параметрам (т.е. по вектору кодирования $\mathbf{y}$), приравняем градиент F^* по $\mathbf{y}$ к нулю и получим следующее уравнение:

$$\sum_{\mathbf{x}} P(\mathbf{X}=\mathbf{x}, \mathbf{Y}=\mathbf{y}) \frac{\partial}{\partial \mathbf{y}} d(\mathbf{x}, \mathbf{y}) = \mathbf{0} \text{ для всех } \mathbf{y} \in \mathbf{Y}, \tag{11.100}$$

где $\mathbf{Y}$ — множество всех векторов кодирования. Используя формулу (11.93) и нормализуя выражение по $P(\mathbf{X} = \mathbf{x})$, можно переопределить условие минимизации следующим образом:

$$\frac{1}{N} \sum_{\mathbf{x}} P(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}) \frac{\partial}{\partial \mathbf{y}} d(\mathbf{x}, \mathbf{y}) = \mathbf{0} \text{ для всех } \mathbf{y} \in \mathbf{Y}, \tag{11.101}$$

где ассоциативная вероятность $P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x})$ определяется из распределения Гиббса (11.97). В выражение (11.101) множитель $1/N$ включен исключительно из соображений последовательности изложения. Здесь N — количество доступных примеров.

Теперь можно описать алгоритм детерминированного отжига для кластеризации [895]. Алгоритм состоит из минимизации Лагранжиана F^* по векторам кодирования при достаточно высокой температуре и тщательным подбором точки минимума при постепенном понижении температуры. Другими словами, детерминированный отжиг работает со специфичным расписанием, в котором температура понижается упорядоченно. При каждом из значений температуры T выполняются два шага, составляющие итерацию алгоритма.

1. Векторы кодирования фиксируются, и для определенной меры искажения $d(\mathbf{x}, \mathbf{y})$ используется распределение Гиббса с целью вычисления ассоциативных вероятностей.
2. Ассоциации фиксируются, и выражение (11.101) используется для оптимизации меры искажения $d(\mathbf{x}, \mathbf{y})$ по векторам кодирования $\mathbf{y}$.

В этой двухшаговой итеративной процедуре функция монотонно не возрастает по F^*, что гарантирует сходимость к минимуму. При высоких значениях температуры T Лагранжиан F^* достаточно гладок и является выпуклой функцией $\mathbf{y}$ (при выполнении мягких допущений относительно меры искажения $d(\mathbf{x}, \mathbf{y})$, сделанных ранее). Глобальный минимум Лагранжиана F^* может быть найден при высоких температурах. По мере понижения температуры T ассоциативные вероятности становятся жесткими, что приводит к жесткому решению задачи кластеризации.

При понижении температуры T в процессе выполнения расписания отжига система проходит через последовательность фазовых переходов, состоящих из естественных разбиений кластеров. При этом размер модели кластеризации увеличивается (т.е. увеличивается количество кластеров) [896], [898]. Это явление имеет важное значение по следующим причинам.

- Оно реализует удобный механизм *управления* размером модели кластеризации.
- При обычном физическом отжиге фазовые переходы являются *критическими точками* (critical point) детерминированного процесса отжига, в которых отжигу должно уделяться большое внимание.
- Критические точки являются *вычисляемыми* (computable), и эта информация может использоваться для ускорения алгоритма между фазовыми переходами.
- С помощью проведения процедуры проверки с последовательностью решений, полученных на различных фазах (являющихся решениями задачи при различных размерах модели), можно получить *модель оптимального размера* (optimum model size).

Пример 11.4

На рис. 11.10, 11.11 показана эволюция кластерного решения, использующего детерминированный отжиг на различных фазах по мере понижения температуры T или увеличении обратной величины $B = 1/T$ [896]. Множество данных, использованных для генерации этих рисунков, состояло из шести множеств с гауссовыми распределениями. Их центры на рисунке отмечены крестиками. Центры вычисленных кластеров на рисунке отмечены кружочками. Так как решения задачи кластеризации при ненулевых температурах не являются жесткими, для получаемых контуров принадлежности к кластеру использовалась одинаковая вероятность (в данном случае — 1/3). Весь процесс начинается с одного естественного кластера, содержащего все множество обучения (см. рис. 11.10, *а*). При первом фазовом переходе он разделяется на два кластера (см. рис. 11.10, *б*), после чего происходит еще несколько фазовых переходов, пока не достигается естественное количество кластеров — шесть. Последующий фазовый переход инициирует "взрыв", при котором все кластеры разбиваются. На рис. 11.11 показана фазовая диаграмма, отображающая значения среднего искажения во время процесса отжига, и количество естественных кластеров на каждой фазе. На этом рисунке среднее искажение (нормализованное по отношению к своему минимальному значению) отображается по отношению к величине, обратной температуре (т.е. B), также нормализованной по отношению к своему минимальному значению. На обеих осях использовался соответствующий логарифмический масштаб. ■

Аналогия с алгоритмом EM

Чтобы описать еще один важный аспект алгоритма детерминированного отжига, рассмотрим ассоциативную вероятность $P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$ как ожидаемое значение двоичной случайной переменной $V_{\mathbf{xy}}$, определенной следующим образом:

$$V_{\mathbf{xy}} = \begin{cases} 1, & \text{если входной вектор } \mathbf{x} \text{ назначен вектору кодирования } \mathbf{y}, \\ 0 - & \text{в противном случае.} \end{cases} \tag{11.102}$$

Исходя из этого, заметим, что двухшаговая итерация алгоритма детерминированного отжига принимает форму алгоритма *ожидания-максимизации* (expectation-maximization — EM) (см. главу 7) оценки максимального правдоподобия. В частности, первый шаг, на котором вычисляются ассоциативные вероятности, является эквивалентом шага ожидания. Второй шаг, на котором минимизируется Лагранжиан F^*, является эквивалентом шага максимизации.

Воспользовавшись такой аналогией, заметим, что детерминированный отжиг является более общим алгоритмом, чем оценка максимального правдоподобия. Это можно утверждать, поскольку, в отличие от оценки максимального правдоподобия, детерминированный отжиг не налагает никаких требований на вид рассматриваемого распределения данных, — минимизируются ассоциативные вероятности, производные от Лагранжиана F^*.

11.14. Резюме и обсуждение

Эта глава посвящена использованию идей статистической механики в качестве математического базиса формулировки методов оптимизации для обучаемых машин. Рассмотренные обучаемые машины можно разбить на две категории.

- *Стохастические машины*, рассмотренные на примерах машины Больцмана, сигмоидальных сетей доверия и машины Гельмгольца.
- *Детерминированные машины*, полученные на основе машин Больцмана и сигмоидальных сетей доверия с помощью введения аппроксимаций среднего поля.

Машина Больцмана использует скрытые и видимые нейроны, представляющие собой стохастические двоичные элементы. Они учитывают прекрасные свойства распределения Гиббса, предлагая, таким образом, следующие привлекательные особенности.

- Посредством обучения распределение вероятности, представляемое нейронами, становится сходным с распределением вероятности среды.
- Эта сеть предлагает обобщенный подход, применимый к основным вопросам поиска, представления и обучения [457].

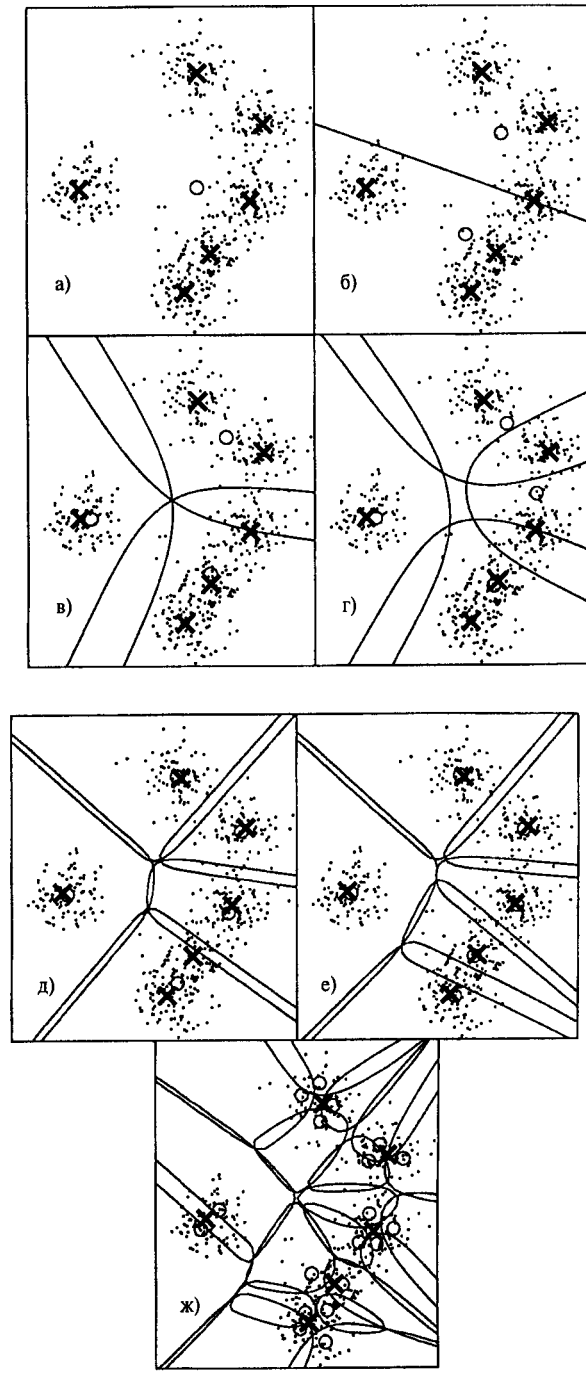


Рис. 11.10. Кластеризация на различных фазах. Линии представляют собой контуры равной вероятности $p = 1/2$ на рисунке (б) и $p = 1/3$ на всех остальных: 1 кластер (B=0) (а); 2 кластера (B=0,0049) (б); 3 кластера (B=0,0056) (в); 4 кластера (B=0,0100) (г); 5 кластеров (B=0,0156) (д); 6 кластеров (B=0,0347) (е); 19 кластеров (B=0,0605) (ж)

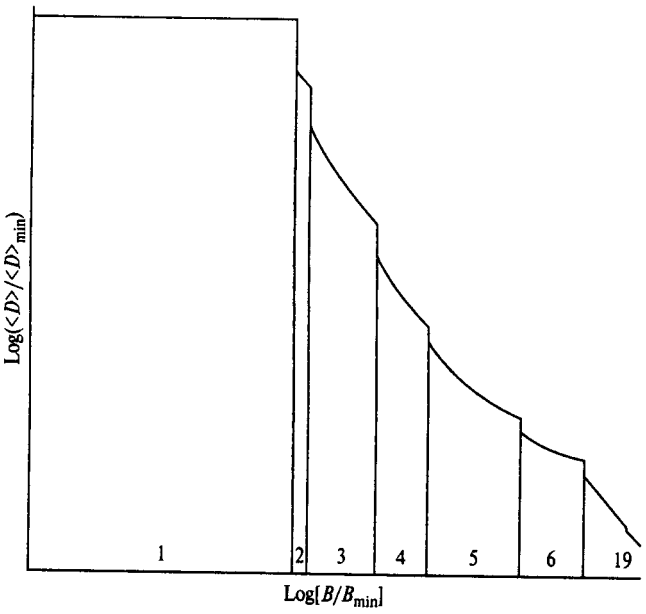


Рис. 11.11. Фазовая диаграмма для примера с несколькими гауссовыми распределениями. Количество эффективных кластеров показано для каждой фазы

- Сеть гарантирует нахождение глобального минимума на поверхности энергии по отношению к *состояниям*, при условии, что расписание отжига в процессе обучения предполагает достаточно медленное уменьшение температуры [343].

К сожалению, допустимое расписание отжига слишком медленно, чтобы иметь какое-то значимое практическое применение. Однако этот процесс обучения может быть ускорен для особых классов машин Больцмана, для которых не требуется запускать алгоритм квантования или применять аппроксимацию среднего поля. В частности, в машинах Больцмана, в которых скрытые нейроны скомпонованы в цепочки, дерево или цепочки деревьев, обучение может достигать полиномиального времени. Это достигается с помощью одного алгоритма стохастической механики, известного под названием *прореживания* (decimation). Это простая и элегантная процедура, которая рекурсивно удаляет связи и узлы из графа, подобно решению *цепи резисторно-индуктивной емкости* (resistance inductance capacitance circuit — RLC) [937], [938].

Сигмоидальные сети доверия предлагают существенное улучшение по сравнению с машинами Больцмана, устраняя потребность в использовании отрицательной фазы алгоритма. Этого они достигают путем замены симметричных связей машины Больцмана направленными ациклическими связями. В то время как машина Больцмана является рекуррентной сетью с избытком обратных связей, сигмоидальные сети доверия имеют многослойную архитектуру с отсутствием обратных связей. Как следует из названия, сигмоидальные сети доверия тесно связаны с классическими сетями доверия, введенными в [822], таким образом, связывая предметную область нейронных сетей с областью *вероятностных рассуждений и графических моделей* (probabilistic reasoning and graphical models) [523], [524].

Машины Гельмгольца представляют собой отдельный класс. Их создание было основано на том, что зрение является обратной графикой [461], [484]. В частности, они используют стохастические порождающие модели, работающие в обратном направлении, для преобразования абстрактных представлений изображений в сами изображения. Сами абстрактные представления изображений (т.е. собственные знания сети об окружающем мире) изучаются стохастической моделью распознавания, работающей в прямом направлении. Посредством интеграции моделей генерации и распознавания (т.е. прямой и обратной проекции) машины Гельмгольца играют роль самообучающейся машины, устраняя тем самым потребность в учителе.

Теперь рассмотрим класс детерминированных машин. Детерминированные машины Больцмана были получены из стохастических посредством применения простейшей формы аппроксимации среднего поля, в которой корреляция между двумя случайными переменными была заменена произведением их средних значений. Это привело к тому, что детерминированная машина Больцмана стала намного быстрее стандартной стохастической. К сожалению, на практике их использование ограничивается одним слоем скрытых нейронов. В [540] было доказано, что при корректном применении теории среднего поля в машине Больцмана корреляция должна вычисляться с помощью *теоремы линейного отклика* (linear response theorem). Сущность этой теоремы состоит в замене фиксированной и свободной корреляций в правиле обучения Больцмана (11.53) их *линейными аппроксимациями отклика* (linear response approximation). Согласно этой работе, новую процедуру обучения можно применять как к сетям со скрытыми нейронами, так и без них.

Детерминированная форма сигмоидальных сетей доверия была получена с помощью другой формы теории среднего поля, в которой строгая форма нижней границы функции логарифмического правдоподобия была выведена с использованием неравенства Йенсена. Более того, эта теория берет на вооружение принципиальные достоинства трактуемой подструктуры, выводя этот класс нейронных сетей в разряд важного дополнения к сетям доверия.

В этой главе также рассматриваются два приема оптимизации: моделирование отжига и детерминированный отжиг. Отличием метода моделирования отжига является его случайное блуждание по поверхности энергии, что может сделать этот метод достаточно медленной процедурой, неприменимой в большинстве приложений. В противоположность этому детерминированный отжиг включает случайность в функцию стоимости, которая затем детерминированным образом оптимизируется, начиная с высокой температуры, которая затем постепенно снижается. Однако следует заметить, что моделирование отжига гарантированно достигает глобального минимума, в то время как это свойство для детерминированного отжига пока еще не доказано.

Несмотря на то что основное внимание в этой главе было сосредоточено на методах оптимизации и стохастических машинах, предназначенных для решения задач обучения без учителя, их можно также использовать по мере необходимости и для задач обучения с учителем.

Задачи

Цепи Маркова

11.1. Вероятность перехода за n шагов из состояния i в состояние j обозначается как $p_{ij}^{(n)}$. Используя метод индукции, покажите, что

$$p_{ij}^{(1+n)} = \sum_k p_{ik} p_{kj}^{(n)}.$$

11.2. На рис. 11.12 показана диаграмма перехода состояний для процесса *случайной прогулки* (random walk), в котором вероятность перехода p больше нуля. Будет ли бесконечно длинная цепь Маркова, показанная здесь, несократимой? Обоснуйте ответ.

11.3. Рассмотрим цепь Маркова, показанную на рис. 11.13. Она несократима. Идентифицируйте классы состояний, содержащиеся на этой диаграмме.

11.4. Вычислите вероятности устойчивых состояний цепи Маркова, показанной на рис. 11.14.

Приемы моделирования

11.5. Алгоритм Метрополиса и квантователь Гиббса представляют собой два альтернативных подхода к моделированию крупных задач. Обсудите основные сходства и различия между ними.

11.6. В этой задаче рассмотрим использование отжига для решения *задачи коммивояжера* (travelling salesman problem). Дано:

- N городов;
- расстояние между всеми парами городов равно d;
- маршрут представляет собой замкнутый путь, содержащий в себе посещения всех городов только по одному разу.

Целью является поиск такого маршрута (т.е. последовательного обхода всех городов), который имеет минимальную общую длину L. В этой задаче различные маршруты представляют собой конфигурации, а минимизации подлежит функция стоимости, которой является длина маршрута.

а) Разработайте итеративный метод генерации допустимых конфигураций.

б) Общая длина маршрута определяется по формуле

$$L_p = \sum_{i=1}^{N} d_{P(i)P(i+1)},$$

где P — перестановка с $P(n+1)$=$P(1)$. Следовательно, функцией разбиения будет следующая:

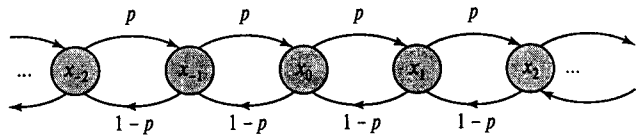


Рис. 11.12. Диаграмма перехода состояний для процесса случайной прогулки

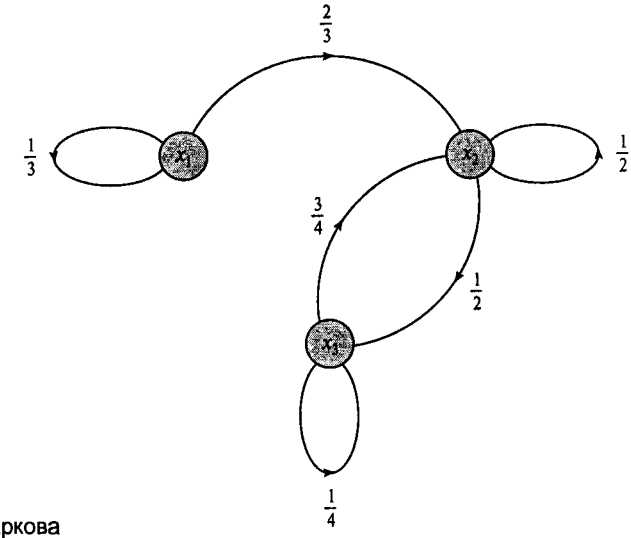


Рис. 11.13. Несократимая цепь Маркова

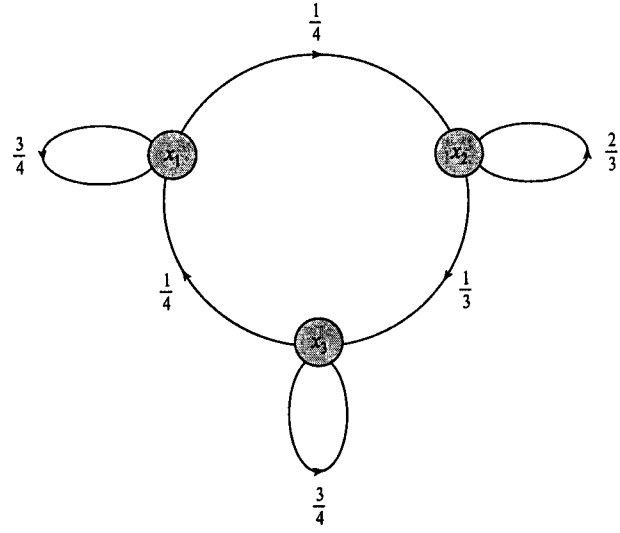


Рис. 11.14. Еще один пример цепи Маркова

$$Z = \sum_{P} e^{-L_P/T}, \tag{11.103}$$

где T — параметр управления. Выведите алгоритм моделирования отжига для описанной задачи коммивояжера.

Машина Больцмана

11.7. Рассмотрим стохастический нейрон j, который может принимать два состояния. Его рабочей температурой является T. Этот нейрон меняет свое состояние с x_j на $-x_j$ с вероятностью

$$P(x_j \to -x_j) = \frac{1}{1 + \exp(-\Delta E_j / T)},$$

где ΔE_j — изменение энергии, возникшее в результате такой смены состояний. Общая энергия машины Больцмана определяется следующей формулой:

$$E = -\frac{1}{2} \sum_i \sum_{j, i \neq j} w_{ji} x_i x_j,$$

где w_{ji} — синаптический вес, ведущий от нейрона i к нейрону j. В машине Больцмана $w_{ji} = w_{ij}$ и $w_{ii} = 0$.

а) Покажите, что

$$\Delta E_j = -2 x_j v_j,$$

где v_j — индуцированное локальное поле нейрона j.

б) Исходя из этого, покажите, что для начального состояния $x_j = -1$ вероятность перехода нейрона j в состояние $+1$ равна $1/(1 + \exp(-2v_j/T))$.

в) Покажите, что формула в п. б сохраняется и для вероятности перехода нейрона j из состояния -1 в состояние $+1$.

11.8. Выведите формулу (11.49), которая определяет производную функции логарифмического правдоподобия $L(\mathbf{w})$ по синаптическим весам w_{ji} в машине Больцмана.

11.9. Распределение Гиббса может быть выведено с помощью полноценного математического подхода, *не полагающегося* на концепции статистической физики. В частности, *модель* стохастической машины в виде *двухшаговой цепи Маркова* может использоваться для формализации допущения, вытекающего из уникальных свойств машины Больцмана [710]. Это не будет неожиданностью, так как моделирование отжига, положенное в основу работы машины Больцмана, само обладает свойством Маркова [1079].

Теперь рассмотрим модель перехода между состояниями нейронов в стохастической машине, состоящей из двух случайных процессов.

- Первый процесс решает, какой переход состояния осуществить.
- Второй процесс решает, принять ли предложение перехода.

а) Выражая вероятность перехода p_{ji} через произведение двух множителей,

$$p_{ji} = \tau_{ji} q_{ji} j \neq i,$$

покажите, что

$$p_{ii} = 1 - \sum_{j \neq i} \tau_{ij} q_{ij}.$$

б) Предполагается, что матрица интенсивности попыток симметрична, т.е.

$$\tau_{ji} = \tau_{ij}.$$

Также предполагается, что вероятность успешности попытки перехода удовлетворяет *свойству дополнения* условной вероятности перехода:

$$q_{ji} = 1 - p_{ij}.$$

Принимая эти два предположения, покажите, что

$$\sum_j \tau_{ji} (q_{ij} \pi_j + q_{ij} \pi_i - \pi_j) = 0.$$

в) Для $\pi \neq 0$ используйте результат п. а настоящей задачи, чтобы показать, что

$$q_{ij} = \frac{1}{1 + (\pi_i / \pi_j)}.$$

г) В заключение выполним замену переменных:

$$E_i = -T \log \pi_i + T^*,$$

где T и T^* — произвольные константы. Исходя из этого, получите следующий результат:

$$\pi_i = \frac{1}{Z}\exp\left(-\frac{E_i}{T}\right),$$
$$Z = \sum_j \exp\left(-\frac{E_j}{T}\right),$$
$$q_{ji} = \frac{1}{1+\exp(-\Delta E/T)},$$

где $\Delta E = E_j - E_i$

д) Какие выводы можно сделать из полученных результатов?

11.10. В разделе 11.7 принцип максимального правдоподобия использовался в качестве критерия для вывода правила обучения Больцмана (11.53). В этой задаче вернемся к этому правилу обучения, но с использованием другого критерия. В главе 10 была определена дивергенция Кулбека–Лейблера между двумя распределениями вероятности p_α^+ и p_α^- в следующем виде:

$$D_{p^+||p^-} = \sum_\alpha p_\alpha^+ \log\left(\frac{p_\alpha^+}{p_\alpha^-}\right),$$

где суммирование выполняется по всем возможным состояниям α. Символ p_α^+ обозначает вероятность того, что видимые нейроны сети находятся в состоянии α в то время, когда сеть находится в фиксированном (положительном) состоянии. Символ p_α^- обозначает вероятность того, что те же нейроны сети находятся в состоянии α в то время, когда сеть находится в свободном (отрицательном) состоянии. С помощью $D_{p^+||p^-}$ выведите правило обучения Больцмана.

11.11. Рассмотрим машину Больцмана, видимые нейроны которой делятся на входные и выходные. Состояния этих нейронов обозначим символами α и γ соответственно. Состояния скрытых нейронов обозначим символом β. Дивергенция Кулбека–Лейблера для этой машины определяется следующей формулой:

$$D_{p^+||p^-} = \sum_\alpha p_\alpha^+ \sum_\gamma p_{\gamma|\alpha}^+ \log\left(\frac{p_{\gamma|\alpha}^+}{p_{\gamma|\alpha}^-}\right),$$

где p_α^+ — вероятность состояния α во всех входных нейронах; $p_{\gamma|\alpha}^+$ — условная вероятность того, что выходные нейроны зафиксированы в состоянии α, если входным состоянием является α; $p_{\gamma|\alpha}^-$ — условная вероятность того, что выходные нейроны находятся в состоянии термального равновесия

γ, при условии, что только входные нейроны находятся в состоянии α. Как и раньше, верхние индексы "плюс" и "минус" обозначают соответственно положительную (фиксированную) и отрицательную (свободную) фазы.

а) Выведите формулу для $D_{p^+||p^-}$ для машины Больцмана, имеющей входные, скрытые и выходные нейроны.

б) Покажите, что правило обучения Больцмана для коррекции синаптических весов w_{ji} в этой конфигурации сети будет выражаться той же формулой, которая описана выражением (11.53), но с новой интерпретацией обозначений p_{ji}^+ и p_{ji}^-.

Сигмоидальные сети доверия

11.12. Обобщите сходства и различия между машиной Больцмана и сигмоидальными сетями доверия.

11.13. В задаче 11.9 показано, что машина Больцмана описывается моделью двухшаговой цепи Маркова. Допускает ли сигмоидальная сеть доверия описание моделью цепи Маркова?

11.14. Пусть w'_{ji} — синаптический вес, идущий от нейрона i к нейрону j в сигмоидальной сети доверия, использующей для включенного и выключенного состояний нейронов значения $+1$ и -1 соответственно. Обозначим w_{ji} соответствующие синаптические веса в сигмоидальной сети доверия, использующей другие значения для состояний — $+1$ и 0 соответственно. Покажите, что веса w_{ji} могут преобразовать в w'_{ji} с помощью следующего преобразования:

$$w'_{ji} = \frac{w_{ji}}{2} \quad \text{при} \quad 0 < i < j,$$

$$w'_{jo} = w_{j0} + \frac{1}{2} \sum_{0<i<j} w_{ji}.$$

В последней строке присутствует порог, примененный к нейрону j.

11.15. В сигмоидальных сетях доверия вероятность $P(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)$ определяется как распределение Гиббса, а вероятность $P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$ — как соответствующая функция разбиения. Обоснуйте справедливость этих двух определений.

Машина Гельмгольца

11.16. Машина Гельмгольца не содержит обратных связей как в модели распознавания, так и в порождающей модели. Что случится, если разрешить использование обратной связи в какой-либо из двух этих моделей.

Детерминированная машина Больцмана

11.17. Машина Больцмана выполняет градиентный спуск (в пространстве весов) по пространству вероятностей (см. задачу 11.10). По какой функции детерминированная машина Больцмана выполняет свой градиентный спуск? Для ответа на этот вопрос можно обратиться к [458].

11.18. Рассмотрим асимметричную рекуррентную сеть (в которой $w_{ji} \neq w_{ij}$). Обсудите, как детерминированный алгоритм обучения Больцмана сможет сделать сеть симметричной, предполагая, что после каждой коррекции каждый из весов приближается к нулю на малую величину, пропорциональную его амплитуде [458].

Детерминированная сигмоидальная сеть доверия

11.19. Покажите, что разность между выражениями в левой и правой частях (11.77) равна дивергенции Кулбека–Лейблера между распределениями $Q(\mathbf{X}_\beta = \mathbf{x}_\beta | \mathbf{X}_\alpha = \mathbf{x}_\alpha)$ и $P(\mathbf{X}_\beta = \mathbf{x}_{\beta|} \mathbf{X}_\alpha = \mathbf{x}_\alpha)$.

11.20. Аргумент сигмоидальной функции (11.89) определяет индуцированное локальное поле v_j нейрона j в детерминированной сигмоидальной сети доверия. Чем это v_j отличается от соответствующего индуцированного локального поля нейрона в многослойном персептроне, обучаемом с помощью алгоритма обратного распространения?

Детерминированный отжиг

11.21. В разделе 11.13 мы рассмотрели идею детерминированного отжига, использующего информационно-теоретический подход. Эта идея детерминированного отжига могла быть разработана также и с использованием принципа максимальной энтропии, описанного в главе 10. Проверьте разумность второго подхода.

11.22. а) Используя выражения (11.97) и (11.98), получите результат, описанный выражением (11.99), в котором определен Лагранжиан F^*, полученный с помощью использования распределения Гиббса для ассоциативной вероятности.

б) Используя результат первой части этой задачи, выведите условие (11.101) для минимума F^* по отношению к вектору кодирования $\mathbf{y}$.

в) Примените условие минимизации (11.101) к мере квадратичного искажения (11.91) и прокомментируйте полученный результат.

11.23. Рассмотрите множество данных, представляющее собой смесь гауссовских распределений. Какие преимущества в такой ситуации может предложить детерминированный отжиг по сравнению с оценкой максимального правдоподобия?

11.24. В этой задаче исследуем использование детерминированного отжига в задаче классификации, решаемой нейронной сетью [734]. Выход нейрона j на выходном слое мы обозначим как $F_j(\mathbf{x})$, где $\mathbf{x}$ — входной вектор. Решение задачи классификации основывается на максимизации дискриминанта $F_j(\mathbf{x})$.

а) В качестве целевой функции рассмотрим следующую:

$$F = \frac{1}{N} \sum_{(\mathbf{x},\mathbf{C})\in\mathbf{T}} \sum_{j} P(\mathbf{x} \in \mathbf{R}_j) F_j(\mathbf{x}),$$

где $\mathbf{T}$ — множество обучения, состоящее из маркированных векторов; $\mathbf{x}$ — входной вектор; $\mathbf{C}$ — его метка класса; $P(\mathbf{x} \in \mathbf{R}_j)$ — вероятность ассоциации между входным вектором $\mathbf{x}$ и классом $\mathbf{R}_j$. Используя принцип максимума энтропии (см. главу 10), сформулируйте распределение Гиббса для $P(\mathbf{x} \in \mathbf{R}_j)$.

б) Обозначим как $<P_e>$ среднюю стоимость ошибки классификации. Выведите формулу Лагранжиана для минимизации $<P_e>$, при условии, что энтропия, соответствующая ассоциативным вероятностям $P(\mathbf{x}\in\mathbf{R}_j)$, равна некоторой константе H.

Глава 12

Нейродинамическое программирование

12.1. Введение

В главе 2 были определены две основные парадигмы обучения — с учителем и без учителя. Вторая парадигма, в свою очередь, подразделяется на самоорганизующееся обучение и обучение с подкреплением. Различные формы обучения с учителем рассматривались в главах 4–7; а виды обучения без учителя — в главах 8–11. В этой главе речь пойдет об *обучении с подкреплением* (reinforcement learning).

Обучение с учителем представляет собой задачу "познания" под руководством учителя. Оно предполагает доступность адекватного множества примеров пар входных и выходных сигналов. В противоположность этому обучение с подкреплением является "поведенческой" задачей. Оно выполняется посредством *взаимодействия* обучаемой системы и среды, в которой система стремится достичь заданной цели, невзирая на наличие неопределенности [100], [1036]. Тот факт, что это обучение осуществляется без учителя, делает обучение с подкреплением особенно привлекательным для динамических ситуаций, где сбор удовлетворительного множества примеров дорог или сложен (если вообще возможен).

Существуют два подхода к обучению с подкреплением[1].

1. *Классический подход* (classical approach), в котором обучение осуществляется методом "кнута и пряника" в процессе чередующихся поощрений и наказаний с целью достижения *высококлассного поведения* (highly skilled behavior).

---

[1] Классический подход к обучению с подкреплением уходит корнями в психологию, а точнее, к ранним работам Торндайка [1052] (посвященным обучению животных) и Павлова [821] (посвященным условному рефлексу). Вклад в развитие обучения с подкреплением также внесла работа, в которой было введено понятие критики (critic) [1140]. Классический подход к обучению с подкреплением рассматривается в [415].

Главный вклад в современный подход к обучению с подкреплением внесли работы, посвященные программе игры в шашки [926], посвященные системам адаптивной критики [100], методам времеиных разностей [1032] и Q-обучению [1115]. В справочнике по интеллектуальному управлению [1130] представлены материалы по оптимальному управлению, обучению с подкреплением и методам адаптивной критики и эвристическому динамическому программированию.

Целиком посвящена современному подходу к обучению с подкреплением [126], а история развития этого вида обучения отслежена в [1036].

2. *Современный подход* (modern approach), который основан на математическом приеме *динамического программирования* (dynamical programming), используемом для формирования последовательности действий с учетом возможных будущих стадий без фактического их осуществления. В этом подходе акцент делается на планирование.

В настоящей главе внимание читателя будет сконцентрировано на современном обучении с подкреплением.

Динамическое программирование[2] представляет собой поэтапный метод принятия решений. Причем перед принятием следующего решения последствия текущего решения предсказываются на некоторый интервал будущего. Ключевым аспектом таких ситуаций является то, что решения не могут приниматься изолированно. Стремление к снижению затрат в настоящем приводится в соответствие с нежелательным увеличением затрат в будущем. Это называется *задачей присваивания коэффициентов доверия* (credit assignment problem), так как каждому решению из множества взаимосвязанных решений либо отпускается кредит доверия, либо оно отвергается. Для оптимального планирования необходимо обеспечить эффективный баланс между текущими и будущими затратами. Достижение такого баланса формализует метод динамического программирования. В частности, динамическое программирование позволяет ответить на следующий вопрос: как система может быть обучена для повышения производительности на длительном промежутке времени, если это может потребовать кратковременного снижения производительности?

Следуя [126], современный подход к обучению с подкреплением будем называть *нейродинамическим программированием* (neurodynamic programming). Для такого выбора есть две причины.

- Теоретической основой этого подхода является метод динамического программирования.
- Обучение выполняется на нейронных сетях.

Краткое определение нейродинамического программирования было предложено в [126].

> *Нейродинамическое программирование позволяет системе обучаться принятию правильных решений с помощью наблюдения за собственным поведением и совершенствовать свое поведение путем использования встроенного механизма усиления.*

Наблюдение за поведением осуществляется с помощью метода моделирования Монте-Карло, реализуемого в автономном режиме. Улучшение поведения посредством подкрепления достигается за счет использования итеративной схемы оптимизации.

---

[2] Методика динамического программирования была разработана Беллманом в конце 1950-х годов [118], [119]. Детальное описание этой области содержится в [125].

Структура главы

Динамическое программирование включает две составляющие: рассматриваемую динамическую систему, функционирующую в дискретном времени, и функцию стоимости (cost), аддитивную по времени. Эти две составляющие рассматриваются в разделе 12.2. В разделе 12.3 последует формулировка уравнения оптимальности Беллмана (Bellman's optimality equation), которое играет важную роль в динамическом программировании. В разделах 12.4 и 12.5 рассматриваются два метода расчета оптимальной стратегии в динамическом программировании — итерации по стратегиям и итерации по значениям.

В разделе 12.6 представлен обзор вопросов, связанных с нейродинамическим программированием. Рассмотрение этих вопросов приводит к изучению приближенной стратегии (approximate police iteration) и Q-обучения, в которых для аппроксимации функций применяются нейронные сети. Эти два алгоритма описываются в разделах 12.7 и 12.8. В разделе 12.9 будет проведено компьютерное моделирование Q-обучения.

Завершает главу раздел 12.10, содержащий выводы и рассуждения.

12.2. Марковский процесс принятия решений

Рассмотрим *обучаемую систему* (learning system) (или (agent)), которая взаимодействует с внешней средой способом, показанным на рис. 12.1, в соответствии с *конечным Марковским процессом принятия решений в дискретном времени* (finite, discrete-time Markovian decision process). Этот процесс характеризуется следующими особенностями.

- Внешняя среда развивается на основе вероятностных законов, принимая конечное множество дискретных состояний. Однако заметим, что эти состояния *не* учитывают прошлых статистик, даже если они могли бы быть полезны обучаемой системе.
- В каждом состоянии существует конечное множество возможных действий, которые может предпринять обучаемая система.
- При выполнении обучаемой системой какого-либо действия взимается определенная плата (стоимость действия).
- Наблюдаемые состояния, совершаемые действия и стоимость действия изменяются в дискретном времени.

В контексте нашей дискуссии *состояние* (state) внешней среды определяется как *совокупность всего опыта, накопленного обучаемой системой в процессе взаимодействия с внешней средой, включающего информацию, необходимую для предсказания*

Рис. 12.1. Блочная диаграмма взаимодействия обучаемой системы со средой

обучаемой системой будущего поведения внешней среды. Случайной переменной X_n обозначим состояние внешней среды на шаге дискретного времени n, а переменной $\mathbf{x}(n)$ — фактическое состояние на шаге n. Все конечное множество состояний обозначим символом $\mathbf{X}$. Неожиданной характерной чертой динамического программирования является то, что оно практически не зависит от природы состояний. Поэтому можно продолжать рассуждения, не делая никаких предположений относительно структуры пространства состояний.

Для некоторого состояния i доступное множество действий (т.е. воздействий, применяемых обучаемой системой к внешней среде) обозначим $\mathbf{A}=\{a_{ik}\}$, где второй индекс, k, в обозначении a_{ik} указывает на то, что при нахождении системы в состоянии i возможно совершение более одного действия в отношении внешней среды. Например, переход внешней среды из состояния i в состояние j при воздействии a_{ik} сам вероятностен по своей природе. Однако *вероятность перехода из состояниі в состояние j целиком зависит от текущего состояния и предпринимаемого действияа$_{ik}$*. Это — *свойство Маркова* (Markov property), о котором речь шла в главе 11. Это свойство критично, так как оно означает, что текущее состояние среды несет в себе всю информацию, необходимую обучаемой системе для принятия решения относительно совершаемого действия.

Случайную переменную, обозначающую действие, предпринимаемое обучаемой системой в момент времени n, обозначим A_n. Пусть $p_{ij}(a)$ обозначает вероятность перехода системы из состояния i в состояние j в ответ на действие, предпринятое на шаге n, где $A_n = a$. Из свойства Маркова имеем:

$$p_{ij}(a) = P(X_{n+1} = j | X_n = i, A_n = a). \tag{12.1}$$

Вероятность перехода $p_{ij}(a)$ удовлетворяет двум основообразующим условиям теории вероятности:

$$p_{ij}(a) \geq 0 \quad \text{для всех } i \text{ и } j, \tag{12.2}$$

$$\sum_j p_{ij}(a) = 1 \quad \text{для всех } i. \tag{12.3}$$

Для заданного количества состояний и заданных вероятностей перехода последовательность состояний внешней среды, возникающих в результате выполнения действий обучаемой системой, формирует *цепь Маркова* (Markov chain) (см. главу 11).

При каждом переходе из одного состояния в другое с обучаемой системы взимается некоторая *плата* (cost), или *стоимость*. Более конкретно, при n-м переходе из состояния i в состояние j под воздействием a_{ik} с обучаемой системы взимается стоимость, обозначаемая $\gamma^n g(i, a_{ik}, j)$, где $g(\cdot, \cdot, \cdot)$ — наперед заданная функция; γ — скаляр из интервала [0, 1), называемый *дисконтирующим множителем* (discount factor). Подстраивая этот множитель, можно управлять окрестностью, которую обучаемая система принимает в расчет при принятии решений. Эта величина определяет отношение долговременной окрестности к кратковременной. В пределе, при $\gamma = 0$, система является "близорукой", т.е. может обозревать только непосредственные следствия своих действий. В дальнейших рассуждениях будем игнорировать это предельное значение, т.е. сократим область определения γ до открытого интервала $(0, 1)$. Если γ достигает значения единицы, будущие затраты становятся более важными в процессе определения оптимального действия.

Интерес представляет формулировка *стратегии* (policy), которая определяется как *отображение состояний в действия*. Другими словами, стратегия является правилом, используемым обучаемой системой для принятия решения относительно того, какое действие предпринять, на основании знаний о текущем состоянии внешней среды. Стратегия обозначается следующим образом:

$$\pi = \{\mu_0, \mu_1, \mu_2, \ldots\}, \tag{12.4}$$

где μ_n — функция отображения состояния $X_n = i$ в действие $A_n = a$ в момент времени n =0, 1, 2, ... Это отображение таково, что

$$\mu_n(i) \in \mathbf{A}_i \text{ для всех состояний } i \in \mathbf{X},$$

где $\mathbf{A}_i$ — множество всех возможных действий, предпринимаемых обучаемой системой в состоянии i. Такие *стратегии* называются *допустимыми* (admissible).

Стратегия может быть стационарной и нестационарной. *Нестационарная* (nonstationary) стратегия зависит от времени (см. (12.4)). Если стратегия от времени не зависит, т.е.

$$\pi = \{\mu, \mu, \mu, \ldots\},$$

она называется *стационарной* (stationary). Другими словами, стационарная стратегия при каждом посещении некоторого состояния определяет одно и то же действие. Для стационарной *стратегии* рассматриваемая цепь Маркова может быть как стационар-

ной, так и *нет* (хотя это не очень умное решение). Если применяется стратегия μ, то последовательность состояний $\{X_n, n = 0, 1, 2, \ldots\}$ формирует цепь Маркова с вероятностями переходов $p_{ij}(\mu(i))$, где $\mu(i)$ обозначает некоторое действие. По этой причине данный процесс получил название *Марковского процесса принятия решений* (Markov decision process).

Постановка задачи

Задача динамического программирования может иметь *конечный* и *бесконечный горизонт* (finite & infinite horizon). В задачах с конечным горизонтом затраты накапливаются за конечное число шагов, в задачах с бесконечным горизонтом — за бесконечное. Задачи с бесконечным горизонтом представляют собой хорошее приближение задач, содержащих конечное, но очень большое количество шагов. Они представляют определенный интерес также из-за того, что дисконты гарантируют конечность затрат всех состояний для любой *стратегии*.

Общие ожидаемые затраты в задачах с бесконечным горизонтом, начинающихся с некоторого состояния $X_0 = i$ и использующих стратегию $\pi = \{\mu_n\}$, определяются по формуле

$$J^{\pi}(i) = E\left[\sum_{n=0}^{\infty} \gamma^n g(X_n, \mu_n(X_n), X_{n+1}) | X_0 = i\right], \tag{12.5}$$

где ожидаемое значение вычисляется по цепи Маркова $\{X_1, X_2, \ldots\}$. Функция $J^{\pi}(i)$ называется *функцией стоимости перехода* (cost-to-go function) для *стратегии* π, начинающейся с состояния i. Ее *оптимальное* значение $J^*(i)$ определяется следующим образом:

$$J^*(i) = \min_{\pi} J^{\pi}(i). \tag{12.6}$$

Если стратегия π стационарна, т.е. $\pi = \{\mu, \mu, \ldots\}$, вместо обозначения $J^{\pi}(i)$ используют $J^{\mu}(i)$ и говорят, что стратегия μ является оптимальной, если

$$J^{\mu}(i) = J^*(i) \text{ для всех начальных состояний } i. \tag{12.7}$$

Теперь можно подытожить постановку задачи динамического программирования следующим образом.

Для данного стационарного Марковского процесса, описывающего взаимодействие обучаемой системы и внешней среды, найти стационарную стратегию $\pi = \{\mu, \mu, \mu, \ldots\}$, *которая минимизирует функцию стоимости перехода* $J^{\mu}(i)$ *для всех начальных состояний* i.

Обратите внимание, что во время обучения поведение обучаемой системы может изменяться во времени. Однако оптимальная стратегия, искомая обучаемой системой, будет стационарной [1115].

12.3. Критерий оптимальности Беллмана

Метод динамического программирования основан на очень простой идее, известной под названием *принципа оптимальности Беллмана* (principle of optimality) [118]. В упрощенном виде этот принцип утверждает следующее [119].

Оптимальная стратегия имеет следующее свойство: какими бы ни были начальное состояние и начальное решение, остальные решения должны составлять оптимальную стратегию по отношению к состоянию, вытекающему из первого решения.

Здесь под термином "решение" понимается один из вариантов управления в конкретный момент времени, а под термином "стратегия" — вся управляющая последовательность или функция.

Для того чтобы сформулировать принцип оптимальности в математических терминах, рассмотрим задачу с конечным горизонтом, для которой функция стоимости перехода имеет следующий вид:

$$J_0(X_0) = E\left[g_K(X_K) + \sum_{n=0}^{K-1} g_n(X_n, \mu_n(X_n), X_{n+1})\right], \tag{12.8}$$

где K — *горизонт* (horizon) (т.е. количество шагов); $g_K(X_K)$ — *терминальная стоимость* (terminal cost). Для данного X_0 математическое ожидание (12.8) формулируется относительно оставшихся состояний X_1, ..., X_{K-1}. С помощью этой терминологии можно формально определить принцип оптимальности Беллмана в следующем виде [125].

Пусть $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{K-1}^*\}$ — *оптимальная стратегия для задачи с конечным горизонтом. Предполагается, что при использовании оптимальной стратегии* π^* *заданное состояние* X_n *наступает с ненулевой вероятностью. Рассмотрим подзадачу, в которой система в момент времени* n *находится в состоянии* X_n, *и предположим, что необходимо минимизировать соответствующую функцию стоимости перехода*

$$J_n(X_n) = E\left[g_K(X_K) + \sum_{k=n}^{K-1} g_k(X_k, \mu_k(X_k), X_{k+1})\right] \tag{12.9}$$

для $n = 0, 1, \ldots, K-1$. *Тогда усеченная стратегия* $\{\mu_n^*, \mu_{n+1}^*, \ldots, \mu_{K-1}^*\}$ *будет оптимальной для этой подзадачи.*

Интуитивно можно понять истинность этого принципа оптимальности. Если бы усеченная стратегия $\{\mu_n^*, \mu_{n+1}^*, \ldots, \mu_{K-1}^*\}$ не была оптимальной, то после достижения в момент времени n состояния X_n можно было бы уменьшить функцию стоимости перехода $J_n(X_n)$, перейдя к стратегии, оптимальной для данной подзадачи.

Приведенный подход к оптимальности основывается на инженерном принципе "разделяй и властвуй". По существу, оптимальная стратегия сложного многоступенчатого планирования или задачи управления строится согласно следующей процедуре.

- Вначале строится оптимальная стратегия для последней подзадачи, включающей в себя только последнюю ступень системы.
- После этого оптимизация расширяется на предыдущую ступень и строится оптимальное решение для последних двух ступеней системы.
- Так продолжается до тех пор, пока вся система не будет охвачена оптимальным решением.

Алгоритм динамического программирования

На основе описанной процедуры можно сформулировать алгоритм динамического программирования, который реализуется в обратном времени, от $N-1$ до 0. Пусть $\pi = \{\mu_1, \mu_2, \ldots, \mu_{K-1}\}$ обозначает некоторую допустимую стратегию. Для каждого $n = 0, 1, \ldots, K-1$ обозначим $\pi^n = \{\mu_n, \mu_{n+1}, \ldots, \mu_{K-1}\}$. Пусть $J_n^*(X_n)$ определяет оптимальные затраты на $(K-n)$-м шаге задачи, который начинается в момент времени n с состояния X_n и завершается в момент времени K, т.е.

$$J_n^*(X_n) = \min_{\pi^n} \underset{(X_{n+1},\ldots,X_{K-1})}{E} \left[g_K(X_K) + \sum_{k=n}^{K-1} g_k(X_k, \mu_k(X_k), X_{k+1}) \right]. \qquad (12.10)$$

Это выражение представляет собой оптимальную форму (12.9). Зная, что $\pi^n = (\mu_n, \pi^{n+1})$, и частично раскрывая сумму в правой части выражения (12.10), можно записать следующее:

$$\begin{aligned} J_n^*(X_n) &= \min_{(\mu_n, \pi^{n+1})} \underset{(X_{n+1},\ldots,X_{K-1})}{E} \Big[g_n(X_n, \mu_n(X_n), X_{n+1}) + g_K(X_K) + \\ &\quad + \sum_{k=n+1}^{K-1} g_k(X_k, \mu_k(X_k), X_{k+1}) \Big] = \\ &= \min_{\mu_n} \underset{X_{n+1}}{E} \Big\{ g_n(X_n, \mu_n(X_n), X_{n+1}) + \qquad (12.11) \\ &\quad + \min_{\pi^n} \underset{(X_{n+2},\ldots,X_{K-1})}{E} \left[g_K(X_K) + \sum_{k=n+1}^{K-1} g_k(X_k, \mu_k(X_k), X_{k+1}) \right] \Big\} = \\ &= \min_{\mu_n} \underset{X_{n+1}}{E} \left[g_n(X_n, \mu_n(X_n), X_{n+1}) + J_{n+1}^*(X_{n+1}) \right], \end{aligned}$$

где в последней строке используется определение (12.10), в котором вместо n подставлено $n+1$. Теперь предположим, что для некоторого n и всех X_{n+1}

$$J^*_{n+1}(X_{n+1}) = J_{n+1}(X_{n+1}). \tag{12.12}$$

Тогда выражение (12.11) можно переписать в следующем виде:

$$J^*_n(X_n) = \min_{\mu_n} \underset{X_{n+1}}{E} \left[g_n(X_n, \mu_n(X_n), X_{n+1}) + J_{n+1}(X_{n+1}) \right]. \tag{12.13}$$

Если соотношение (12.12) выполняется для всех X_{n+1}, то равенство

$$J^*_n(X_n) = J_n(X_n)$$

также выполняется для всех X_n. Следовательно, из (12.13) можно заключить, что

$$J_n(X_n) = \min_{\mu_n} \underset{X_{n+1}}{E} \left[g_n(X_n, \mu_n(X_n), X_{n+1}) + J_{n+1}(X_{n+1}) \right].$$

Таким образом, можно формально записать алгоритм динамического программирования в следующем виде [125].

Для любого исходного состояния X_0 оптимальное значение функции стоимости $J^*(X_0)$ равно $J_0(X_0)$, где функция J_0 вычисляется начиная с последнего шага следующего алгоритма:

$$J_n(X_n) = \min_{\mu_n} \underset{X_{n+1}}{E} \left[g_n(X_n, \mu_n(X_n), X_{n+1}) + J_{n+1}(X_{n+1}) \right], \tag{12.14}$$

который "перемещается" по оси времени в обратном направлении, при этом

$$J_K(X_K) = g_K(X_K).$$

Более того, если значение μ^*_n минимизирует правую часть (12.14) для всех X_n и n, тогда стратегия $\pi^* = \{\mu^*_0, \mu^*_1, \ldots, \mu^*_{K-1}\}$ является оптимальной.

Уравнение оптимальности Беллмана

Основная форма алгоритма динамического программирования связана с решением задачи с конечным горизонтом. Однако этот алгоритм можно расширить для решения задач с бесконечным горизонтом, которые описываются функцией стоимости (12.5) при стационарной стратегии $\pi = \{\mu, \mu, \mu, \ldots\}$. Для достижения этой цели нам нужно сделать следующее.

- Изменить индекс времени в алгоритме так, чтобы он соответствовал задаче.
- Определить стоимость $g_n(X_n, \mu(X_n), X_{n+1})$ как

$$g_n(X_n, \mu(X_n), X_{n+1}) = \gamma^n g(X_n, \mu(X_n), X_{n+1}). \tag{12.15}$$

Теперь алгоритм динамического программирования можно переформулировать в следующем виде (см. задачу 12.4):

$$J_{n+1}(X_0) = \min_{\mu} \underset{X_1}{E} [g(X_0, \mu(X_0), X_1) + \gamma J_n(X_1)]. \tag{12.16}$$

Этот алгоритм начинается с исходного условия

$$J_0(X) = 0 \text{ для всех } X.$$

Итак, алгоритм начинается с некоторого исходного состояния X_0; новое состояние X_1 получается в результате применения стратегии μ, а γ в (2.15) представляет собой дисконтный множитель.

Пусть $J^*(i)$ означает оптимальную стоимость в задаче с бесконечным горизонтом и начальным состоянием $X_0 = i$. Тогда $J^*(i)$ можно рассматривать как предел соответствующей оптимальной стоимости $J_K(i)$ на K-м шаге при K, стремящемся к бесконечности:

$$J^*(i) = \lim_{K \to \infty} J_K(i) \text{ для всех } i. \tag{12.17}$$

Это соотношение является связующим звеном между дисконтированными задачами с конечным и бесконечным горизонтами. Принимая $n+1 = K$ и $X_0 = i$ в (12.16), а затем применяя (12.17), получим:

$$J^*(i) = \min_{\mu} \underset{X_1}{E} [g(i, \mu(i), X_1) + \gamma J^*(X_1)]. \tag{12.18}$$

Для оценки оптимальных затрат $J^*(i)$ в задаче с бесконечным горизонтом выполняются следующие действия.

1. Вычисляем математическое ожидание стоимости $g(i, \mu(i), X_1)$ по X_1:

$$E[g(i), \mu(i), X_1] = \sum_{j=1}^{N} p_{ij} g(i, \mu(i), j), \tag{12.19}$$

где N — количество состояний среды; p_{ij} — вероятность перехода из начального состояния $X_0 = i$ в новое состояние $X_1 = j$. Величина, определяемая выражением (12.19), называется *мгновенной ожидаемой стоимостью* (immediate expected

cost) в состоянии i, при выполнении действия, рекомендованного стратегией μ. Обозначая эти затраты как $c(i, \mu(i))$, можно записать, что

$$c(i, \mu(i)) = \sum_{j=1}^{N} p_{ij} g(i, \mu(i), j). \tag{12.20}$$

2. Оцениваем ожидание $J^*(X_1)$ по отношению к X_1. Отсюда можно заключить, что в системе с конечным числом состояний, если известна стоимость перехода $J^*(X_1)$ для всех состояний X_1, можно наверняка определить ожидание $J^*(X_1)$ в терминах вероятностей перехода соответствующей цепи Маркова, записав:

$$E[J^*(X_1)] = \sum_{j=1}^{N} p_{ij} J^*(j). \tag{12.21}$$

Таким образом, подставляя выражения (12.19)–(12.21) в (12.16), получим искомый результат:

$$J^*(i) = \min_{\mu} \left(c(i, \mu(i)) + \gamma \sum_{j=1}^{N} p_{ij}(\mu) J^*(j) \right), i = 1, 2, \ldots, N. \tag{12.22}$$

Равенство (12.22) называется *уравнением оптимальности Беллмана* (Bellman's optimality equation). Его *нельзя* рассматривать как алгоритм — оно представляет собой систему N уравнений, в которой для каждого состояния отводится одна переменная. Решение этой системы уравнений определяет оптимальные значения *функции стоимости перехода* (cost-to-go function) для всех N состояний внешней среды.

Существуют два основных метода вычисления оптимальной стратегии: итерация по стратегиям и итерация по значениям. Эти два метода описываются в разделах 12.4 и 12.5.

12.4. Итерация по стратегиям

Чтобы подготовить основу для описания алгоритма итерации по стратегиям, введем понятие Q-фактора [1115]. Рассмотрим существующую стратегию μ, для которой функция стоимости перехода $J^{\mu}(i)$ известна для всех состояний i. *Q-фактор* (Q-factor) каждой пары состояния $i \in \mathbf{X}$ и действия $a \in \mathbf{A}_i$ определяется как *мгновенная стоимость плюс сумма дисконтированных стоимостей всех последующих состояний согласно стратегии* μ, т.е.

$$Q^{\mu}(i, a) = c(i, a) + \gamma \sum_{j=1}^{n} p_{ij}(a) J^{\mu}(j), \tag{12.23}$$

где действие $a = \mu(i)$. Заметим, что Q-факторы $Q^{\mu}(i,a)$ содержат больше информации, чем функция стоимости перехода $J^{\mu}(i)$. Например, действия можно упорядочить только на основании Q-фактора, в то время как упорядочивание на основе функции стоимости перехода требует знания вероятностей и стоимости перехода между состояниями.

Для более глубокого изучения Q-фактора рассмотрим новую систему, состояния которой образованы на основе исходных состояний 1, 2, ..., N и всех возможных пар (i,a) (рис. 12.2). Здесь возможны две ситуации.

- Система находится в состоянии (i,a), в котором не предпринимается никаких действий. Переход автоматически выполняется в состояние j с вероятностью, скажем, $p_{ij}(a)$, при этом автоматически вычисляется стоимость $g(i,a,j)$.
- Система находится, к примеру, в состоянии i, в котором предпринимается действие $a \in \mathbf{A}_i$. Следующее состояние предопределено как (i,a).

Стратегия μ называется *жадной* (greedy) по отношению к функции стоимости перехода $J^{\mu}(i)$, если для всех состояний $\mu(i)$ представляет собой действие, удовлетворяющее следующему условию:

$$Q^{\mu}(i,\mu(i)) = \min_{a\in\mathbf{A}_i} Q^{\mu}(i,a) \text{ для всех } i. \tag{12.24}$$

Здесь важно обратить внимание на следующие моменты.

- В некоторых состояниях возможно несколько действий, которые минимизируют множество Q-факторов. В этом случае может существовать более одной жадной стратегии по отношению к соответствующей функции стоимости перехода.
- Некоторая стратегия может оказаться жадной по отношению к нескольким различным функциям стоимости перехода.

Более того, следующий факт является основой всех методов динамического программирования:

$$Q^{\mu^*}(i,\mu^*(i)) = \min_{a\in\mathbf{A}_i} Q^{\mu^*}(i,a), \tag{12.25}$$

где μ^* — оптимальная стратегия; J^* — соответствующая функция стоимости перехода.

Имея в распоряжении понятия Q-фактора и жадной стратегии, можно описать *алгоритм итерации по стратегиям* (policy iteration algorithm). Этот алгоритм включает следующие шаги [125].

1. *Шаг вычисления стратегии* (policy evaluation step), на котором определяются функция стоимости перехода для некоторой текущей стратегии и соответствующий Q-фактор.

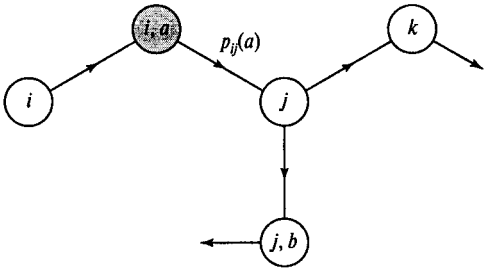


Рис. 12.2. Два возможных перехода: переход из состояния (i, a) в состояние j является вероятностным, а переход из состояния i в состояние (i, a) является детерминистским

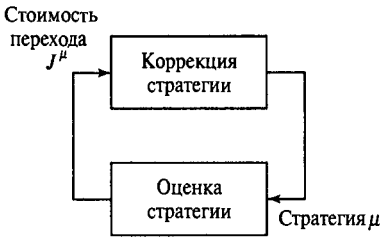


Рис. 12.3. Блочная диаграмма алгоритма итерации по стратегиям

2. *Шаг улучшения стратегии* (policy improvement step), в котором текущая стратегия корректируется и становится жадной по отношению к функции стоимости перехода, полученной на шаге 1.

Эти два шага показаны на рис. 12.3. Для большей конкретизации начнем с некоторой исходной стратегии μ_0, после чего сгенерируем последовательность новых стратегий $\mu_1, \mu_2, \ldots$. Для данной стратегии μ_n реализуем шаг вычисления стратегии, получив функцию стоимости перехода $J^{\mu_n}(i)$ в качестве решения линейной системы уравнений (12.22):

$$J^{\mu_n}(i) = c(i, \mu_n(i)) + \gamma \sum_{j=1}^{N} p_{ij}(\mu_n) J^{\mu_n}(j), \quad i = 1, 2, \ldots, N \tag{12.26}$$

относительно неизвестных $J^{\mu_n}(1), J^{\mu_n}(2), \ldots, J^{\mu_n}(N)$. Используя полученные результаты, можно вычислить Q-фактор для пары состояние–действие (i, a) (см. (12.23)):

$$Q^{\mu_n}(i, a) = c(i, a) + \gamma \sum_{j=1}^{n} p_{ij}(a) J^{\mu_n}(j), \quad a \in \mathbf{A}_i \text{ и } i = 1, 2, \ldots, N. \tag{12.27}$$

Далее выполняем шаг улучшения стратегии, вычисляя новую стратегию μ_{n+1} следующим образом (см. (12.24)):

$$\mu_{n+1}(i) = \arg \min_{a \in \mathbf{A}_i} Q^{\mu_n}(i, a), \quad i = 1, 2, \ldots, N. \tag{12.28}$$

ТАБЛИЦА 12.1. Алгоритм итерации по стратегиям

1.	Начинаем с произвольной исходной стратегии μ_0.
2.	Для $n = 0, 1, 2, \ldots$ вычисляем $J^{\mu_n}(i)$ и $Q^{\mu_n}(i, a)$ для всех состояний $i \in \mathbf{X}$ и действий $a \in \mathbf{A}_i$.
3.	Для каждого из состояний i вычисляем $\mu_{n+1}(i) = \arg\min_{a \in \mathbf{A}_i} Q^{\mu_n}(i, a)$.
4.	Повторяем действия 2, 3, пока μ_{n+1} не перестанет улучшать μ_n. В этой точке алгоритм останавливается, а μ_n считается искомой стратегией.

Затем описанный выше двухшаговый процесс повторяется для стратегии μ_{n+1}, пока мы не получим

$$J^{\mu_{n+1}}(i) = J^{\mu_n}(i) \quad \text{для всех} \quad i.$$

В этом случае алгоритм завершает работу со стратегией μ_n. Если $J^{\mu_{n+1}} \leq J^{\mu_n}$ (см. задачу 12.5), то можно сказать, что алгоритм итерации по стратегиям завершится после конечного количества итераций, так как соответствующий *Марковский процесс принятия решений* (Markovian decision process) имеет конечное количество состояний. В табл. 12.1 в сжатом виде представлен алгоритм итерации по стратегиям, основанный на выражениях (12.26)–(12.28).

12.5. Итерация по значениям

В алгоритме итерации по стратегиям функция стоимости перехода должна была заново вычисляться на каждом шаге алгоритма. С точки зрения вычислений это слишком затратный подход. Даже если функция стоимости перехода для новой стратегии является идентичной соответствующей функции для старой стратегии, данные вычисления не уменьшаются. Однако существует другой метод нахождения оптимальной стратегии, в котором обременительное действие постоянного вычисления функции стоимости перехода отсутствует. Этот альтернативный метод, основанный на последовательных аппроксимациях, называется алгоритмом итерации по значениям.

Алгоритм итерации по значениям (value iteration algorithm) подразумевает решение уравнения оптимальности Беллмана (12.22) для последовательности задач с конечным горизонтом. При устремлении количества итераций алгоритма к бесконечности функция стоимости перехода задачи с конечным горизонтом равномерно сходится по состояниям к соответствующей функции стоимости задачи с бесконечным горизонтом [125], [905].

Пусть $J_n(i)$ — функция стоимости перехода для состояния i на шаге n алгоритма итерации по значениям. Этот алгоритм начинается с произвольного решения $J_0(i)$ для $i = 1, 2, \ldots, N$. Единственным ограничением, налагаемым на $J_0(i)$, является ее ограниченность. Это условие автоматически выполняется в задачах с конечным числом состояний. Если доступна некоторая оценка оптимальной функции стоимости перехода $J^*(i)$, то ее можно использовать в качестве начального значения $J_0(i)$. Как только выбрана функция $J_0(i)$, можно вычислить последовательность функций $J_1(i), J_2(i), \ldots$, используя алгоритм итерации по значениям:

$$J_{n+1}(i) = \min_{a \in \mathbf{A}_i} \left\{ c(i,a) + \gamma \sum_{j=1}^{N} p_{ij}(a) J_n(j) \right\}, \quad i = 1, 2, \ldots, N. \tag{12.29}$$

Применение модификации функции стоимости перехода, описываемой выражением (12.29) для состояния i, называется *резервированием затрат* (backing up of it's cost). Это резервирование является прямой реализацией уравнения оптимальности Беллмана (12.22). Обратите внимание, что значения функции стоимости перехода (12.29) для состояний $i = 1, 2, \ldots, N$ резервируются одновременно на каждой итерации алгоритма. Этот метод реализации представляет собой традиционную *синхронную* форму алгоритма итерации по значениям[3]. Таким образом, начиная с произвольных исходных значений $J_0(1)$, $J_0(2)$, ..., $J_0(N)$, алгоритм (12.29) сходится к оптимальным значениям $J^*(1), J^*(2), \ldots, J^*(N)$ при стремлении количества итераций к бесконечности [125], [905].

В отличие от алгоритма итерации по стратегиям в алгоритме итерации по значениям оптимальная стратегия не вычисляется напрямую. Вместо этого сначала с помощью (12.29) вычисляются оптимальные значения $J^*(1), J^*(2), \ldots, J^*(N)$. Затем по этому оптимальному множеству вычисляется жадная стратегия, т.е.

$$\mu^*(i) = \arg \min_{a \in \mathbf{A}_i} Q^*(i,a), \quad i = 1, 2, \ldots, N, \tag{12.30}$$

где

$$Q^*(i,a) = c(i,a) + \gamma \sum_{j=1}^{n} p_{ij}(a) J^*(j), \quad i = 1, 2, \ldots, N. \tag{12.31}$$

В табл. 12.2 приведено пошаговое описание алгоритма итерации по значениям, основанное на формулах (12.29)–(12.31). В это описание включен критерий останова алгоритма.

---

[3] Итерация по стратегиям и итерация по значениям — два основных метода динамического программирования. Существуют еще два метода дииамического программироваиия, которые нельзя не упомянуть, — метод Гаусса–Зейделя и асинхронное динамическое программирование [99], [125]. В методе Гаусса–Зейделя фуикция стоимости перехода корректируется в каждый момеит времени только для одного состояния; при этом все остальные состояния разворачиваются (sweep). Конкуренция основывается на самых последиих значеииях состояиий. Асинхрониое динамическое программирование отличается от метода Гаусса–Зейделя тем, что оио не организовано в терминах систематических последовательных разверток множества состояний.

ТАБЛИЦА 12.2. Алгоритм итерации по значениям

1. Начинаем с произвольного начального значения $J_0(i)$ для состояний $i = 1, 2, \ldots, N$.
2. Для $n = 0, 1, 2, \ldots$ вычисляем
$$J_{n+1}(i) = \min_{a \in \mathbf{A}_i} \left\{ c(i,a) + \gamma \sum_{j=1}^{N} p_{ij}(a) J_n(j) \right\}, \quad a \in \mathbf{A}_i, \quad i = 1, 2, \ldots, N.$$
Эти вычисления продолжаются, пока не выполнится следующее неравенство:
$$|J_{n+1}(i) - J_n(i)| < \varepsilon \text{ для всех состояний } i,$$
где ε — некоторый наперед заданный параметр допуска. Для того чтобы функция $J_n(i)$ хорошо приближала оптимальную функцию стоимости перехода $J^*(i)$, значение ε должно быть достаточно малым. Таким образом, можно положить:
$$J_n(i) = J^*(i) \text{ для всех состояний } i.$$
3. Вычисляем Q-фактор:
$$Q^*(i,a) = c(i,a) + \gamma \sum_{j=1}^{n} p_{ij}(a) J^*(j), \quad a \in \mathbf{A}_i, \quad i = 1, 2, \ldots, N.$$
Тогда в качестве оптимальной можно выбрать жадную стратегию для $J^*(i)$:
$$\mu^*(i) = \arg \min_{a \in \mathbf{A}_i} Q^*(i,a).$$

Пример 12.1

Задача дилижанса

Для того чтобы проиллюстрировать полезность Q-фактора в динамическом программировании, рассмотрим *задачу дилижанса* (stagecoach problem). В середине XIX века, когда Калифорнию охватила золотая лихорадка, некий авантюрист отправляется на запад Америки [456]. Это путешествие предполагает переезд на дилижансах по неосвоенным землям, где велика опасность нападений бандитов. Исходный пункт путешествия (Миссури) и пункт назначения (Калифорния) фиксированы, однако предстоит выбор маршрута путешествия по восьми промежуточным штатам (рис. 12.4). На этом рисунке показано следующее.

- Общее количество штатов (10); каждый штат представлен соответствующей буквой.
- Направление движения — слева направо.
- В пути от исходного штата A (Миссури) в штат назначения J (Калифорния) нужно сменить четыре дилижанса.
- При перемещении из одного штата в следующий путник может выбрать одно из следующих направлений: вверх, прямо и вниз.
- Существует 18 возможных маршрутов перемещения из штата A в штат J.

На рис. 12.4 также показаны затраты, связанные с соответствующими стратегиями страхования жизни при перемещении на определенных дилижансах, основанные на оценке риска подобной поездки. Требуется найти маршрут из штата A в штат J с самой дешевой стратегией страхования.

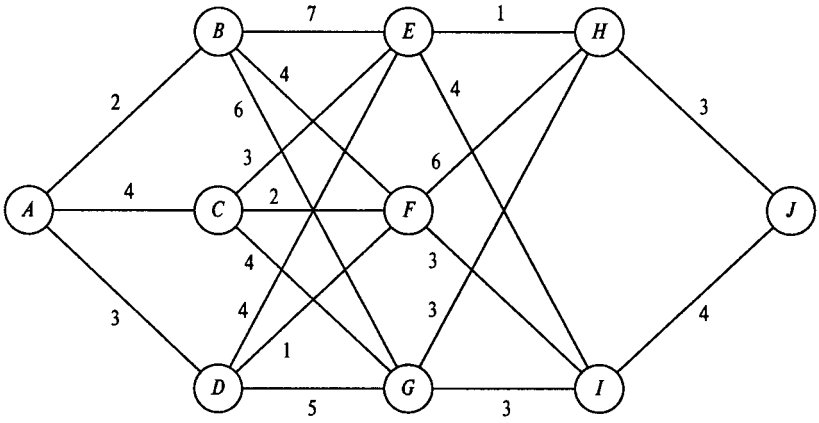


Рис. 12.4. Граф передачи сигнала для задачи дилижанса

Для того чтобы найти оптимальный маршрут, рассмотрим последовательность задач с конечным горизонтом, начинающихся в конечной точке J и продвигающихся в направлении, обратном направлению маршрута. Это соответствует принципу оптимальности Беллмана, описанному в разделе 12.3.

На рис. 12.5, *а* видно, что Q-факторы для последнего переезда к пункту назначения равны:

$Q(H$, вниз)=3,

$Q(I$, вверх)=4.

Эти числа на рис. 12.5, *а* можно увидеть над условным обозначением штатов H и I.

Далее, перемещаясь на один шаг назад и используя значения из рис. 12.5, а, находим следующие Q-факторы:

$Q(E$, прямо)=1+3=4,

$Q(E$, вниз)=4+4=8,

$Q(F$, вверх)=6+3=9,

$Q(F$, вниз)=3+4=7,

$Q(G$, вверх)=3+3=6,

$Q(G$, прямо)=3+4=7.

Так как требуется найти маршрут с наименьшей стратегией страхования, то по значениям Q-факторов видно, что необходимо оставить только следующие маршруты $E \rightarrow H, F \rightarrow I$ и $G \rightarrow H$, а остальные отбросить (рис. 12.5, *б*).

Переместившись назад еще на один пункт, повторив вычисления Q-факторов для состояний B, C, D и оставив только значения, соответствующие наименьшим стратегиям страхования, получим рис. 12.5, *в*.

В заключение, переместившись еще на один шаг и действуя аналогичным способом, получим граф, показанный на рис. 12.5, *г*. На этом рисунке показано, что существуют три оптимальных маршрута:

$$A \rightarrow C \rightarrow E \rightarrow H \rightarrow J,$$

$$A \rightarrow D \rightarrow E \rightarrow H \rightarrow J,$$

$$A \rightarrow D \rightarrow F \rightarrow I \rightarrow J.$$

Все они предполагают затраты, равные 11 условным единицам. ■

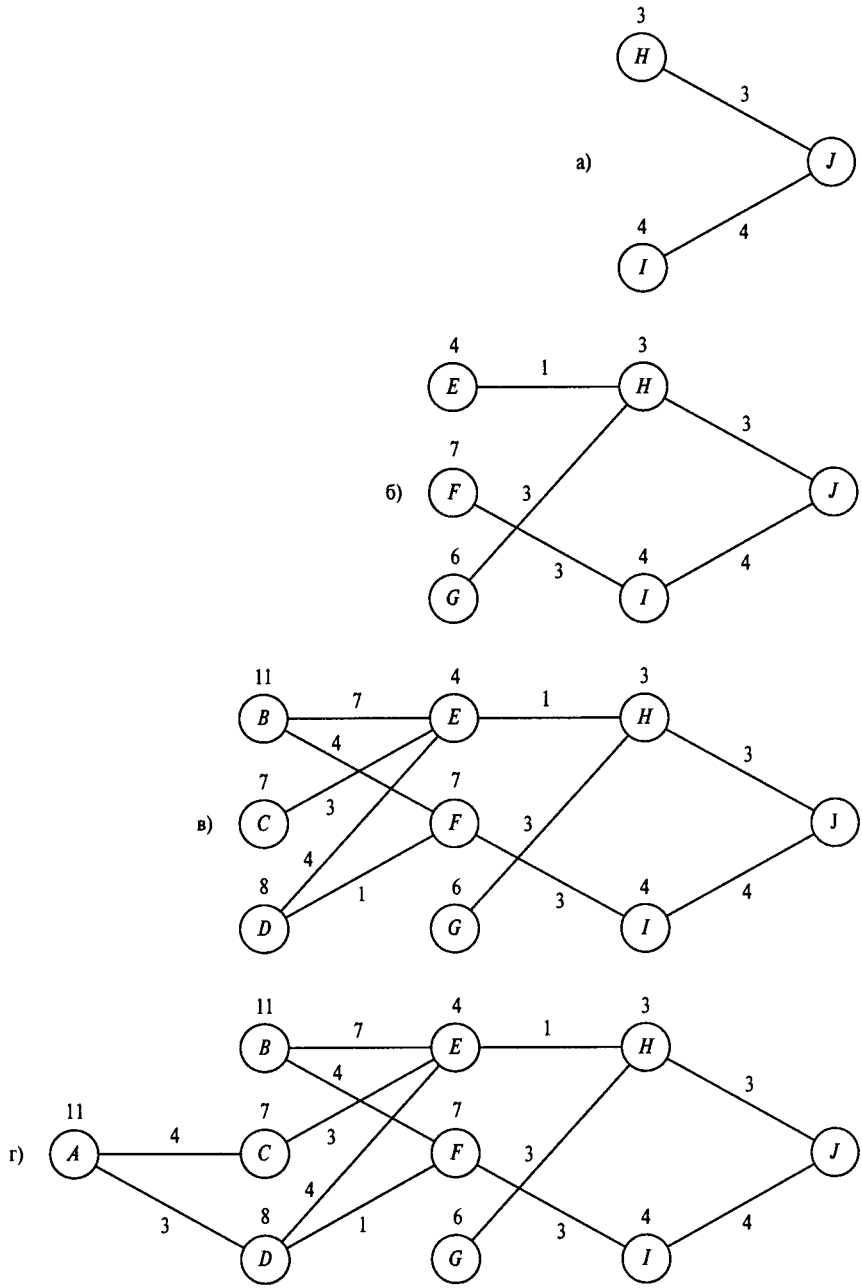


Рис. 12.5. Шаги вычисления Q-факторов в рассматриваемой задаче дилижанса

12.6. Нейродинамическое программирование

Основной целью динамического программирования является поиск оптимальной стратегии, т.е. оптимальной последовательности действий, которые должны быть предприняты обучаемой системой в каждом из ее состояний. В этом контексте существуют два практических вопроса, которые нужно учитывать при выборе алгоритмов итерации по стратегиям и итерации по значениям для решения задачи динамического программирования.

- *Проклятие размерности* (curse of dimensionality). Во многих сложных реальных задачах количество возможных состояний и действий является настолько большим, что требования к вычислительным мощностям реализации алгоритма динамического программирования становятся просто нереальными. Для задачи динамического программирования, содержащей N возможных состояний и M возможных действий для каждого состояния, каждая итерация алгоритма требует N^2M операций для стационарной стратегии. Это часто делает невозможным выполнение даже одной итерации алгоритма, если N достаточно велико. Например, при игре в нарды (backgammon) имеется 10^{20} возможных состояний. Это значит, что одна итерация алгоритма на среднем компьютере, осуществляющем 1 миллион операций в секунду, может занять около 1000 лет [99].
- *Неполнота информации* (incomplete information). Алгоритмы итераций по стратегиям и по значениям требуют наличия априорных знаний о соответствующем Марковском процессе принятия решений. Это значит, что для вычисления оптимальной стратегии требуется знать вероятности перехода между состояниями p_{ij} и наблюдаемые стоимости $g(i, a, j)$. К сожалению, априорные знания не всегда доступны.

В свете любой из этих сложностей иногда приходится отказаться от поиска оптимальной стратегии и искать *субоптимальную*.

В контексте данной книги интерес представляет субоптимальная процедура, которая включает в себя использование нейронных сетей и (или) моделирование с целью аппроксимации функции стоимости перехода $J^*(i)$ для всех $i \in \mathbf{X}$. В частности, для конкретного состояния i функция $J^*(i)$ заменяется подходящей аппроксимацией $\hat{J}(i, \mathbf{w})$, где $\mathbf{w}$ — вектор параметров. Функция $\hat{J}(\cdot, \mathbf{w})$ называется *функцией отсчета* (score function) или *приближенной функцией стоимости перехода* (approximate cost-to-go function). Сами значения $\hat{J}(i, \mathbf{w})$ называются *отсчетами* или *приближенными затратами* для состояния i. Таким образом, отсчет $\hat{J}(i, \mathbf{w})$ является откликом нейронной сети в ответ на подачу на вход состояния i. Здесь нейронные сети функционируют в качестве *универсальных аппроксиматоров*. Это одно из встроенных свойств многослойных персептронов и сетей на основе радиальных базисных функций.

Нас интересуют те задачи динамического программирования, которые имеют большое количество состояний и в которых требуется найти такую функцию от-

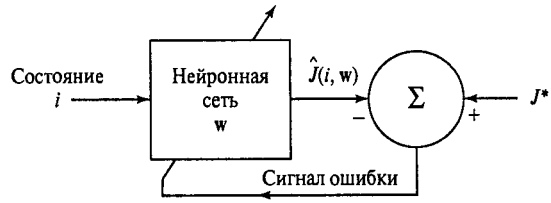


Рис. 12.6. Нейронная сеть, предназначенная для аппроксимации оптимальной функции стоимости перехода J^*

счета $\hat{J}(\cdot, \mathbf{w})$, для которой вектор параметров $\mathbf{w}$ имеет небольшую размерность. В этой форме аппроксимации, называемой *компактным представлением* (compact representation), сохраняются только вектор параметров $\mathbf{w}$ и общая структура функции отсчета $\hat{J}(\cdot, \mathbf{w})$. Отчеты $\hat{J}(i, \mathbf{w})$ для всех состояний $i \in \mathbf{X}$ генерируются только по мере необходимости. Требуется алгоритмически найти вектор параметров $\mathbf{w}$, такой, чтобы для данной структуры нейронной сети (например, для многослойного персептрона) отсчет $\hat{J}(i, \mathbf{w})$ предоставлял удовлетворительную аппроксимацию оптимального значения $J^*(i)$ для всех $i \in \mathbf{X}$.

Из материала по обучению с учителем, представленного в главах 4–7, известно, что для обучения нейронной сети, независимо от ее типа, требуется наличие множества маркированных данных, являющегося представительным для своей задачи. Однако в контексте задач динамического программирования множества данных (т.е. пар примеров "вход-выход" $\{(i, J^*(i))\}$), необходимых для оптимизации структуры сети в некотором статистическом смысле, не существует (рис. 12.6). Остается лишь возможность использования метода *моделирования Монте-Карло* (Monte Carlo simulation), в которой вместо фактической системы, характеризуемой Марковским процессом принятия решений, используется некоторая суррогатная модель. В результате формируется автономный режим работы процесса динамического программирования, который обладает следующими потенциальными преимуществами [126].

1. Использование моделирования для приближенной оценки оптимальной функции стоимости перехода является ключевой идеей, которая отличает методологию нейродинамического программирования от традиционных методов аппроксимации в динамическом программировании.
2. Моделирование позволяет использовать методы нейродинамического программирования для создания систем, точных моделей которых не существует. Для таких систем традиционные приемы динамического программирования неприменимы (например, если невозможно оценить вероятности состояний и переходов).
3. Посредством моделирования можно идентифицировать наиболее важные или наиболее представительные состояния системы, например те, которые при моделировании посещаются чаще других. Следовательно, функция отсчета, созданная нейронной сетью, будет являться хорошей аппроксимацией оптимальной функции *стоимости перехода* для этих конкретных состояний. В итоге можно будет выработать хорошую субоптимальную стратегию для сложных задач динамического программирования.

Однако важно помнить что при использовании аппроксимации нельзя ожидать сходимости функции отсчета $\hat{J}(\cdot, \mathbf{w})$ к оптимальной функции стоимости перехода. Причина этого заключается в том, что $J^*(\cdot)$ может не принадлежать множеству функций, которые могут быть точно представлены выбранной структурой нейронной сети.

В следующих двух разделах мы рассмотрим две процедуры приближенного динамического программирования, использующие аппроксимации функций стоимости перехода. Первая процедура, описанная в разделе 12.7, связана с приближением алгоритма итерации по стратегиям, предполагающего доступность Марковской модели системы. Вторая процедура, рассматриваемая в разделе 12.8, связана с так называемым Q-обучением. Она требует достаточно мягких предположений.

12.7. Приближенный алгоритм итерации по стратегиям

Предположим, что существует некоторая задача динамического программирования с довольно большим множеством допустимых состояний и действий, что делает непрактичным применение традиционных подходов. Пусть имеется модель системы, т.е. известны вероятности переходов $p_{ij}(a)$ и наблюдаемые затраты $g(i, a, j)$. Для реализации алгоритма итерации по стратегиям предлагается использовать аппроксимацию, основанную на моделировании Монте-Карло и методе наименьших квадратов, что и будет описано далее [125].

На рис. 12.7 показана упрощенная блочная диаграмма *приближенного алгоритма итерации по стратегиям* (approximate policy iteration algorithm). Она аналогична диаграмме обычного алгоритма итерации по стратегиям, представленной на рис. 12.3, но имеет одно важное отличие: шаг вычисления точной оценки стратегии заменен вычислением *приближенной* оценкой. Таким образом, приближенный алгоритм итерации по стратегиям чередует два шага: шаг оценки приближенной стратегии и шаг улучшения стратегии.

1. *Шаг вычисления приближенной стратегии* (approximate policy evaluation step). Для данной стратегии μ вычисляется приближенная функция стоимости перехода $\hat{J}^{\mu}(i, \mathbf{w})$ для всех состояний i. Вектор $\mathbf{w}$ является вектором параметров нейронной сети, используемой для аппроксимации функции.
2. *Шаг улучшения стратегии* (policy improvement step). С использованием приближенной функции стоимости перехода $\hat{J}^{\mu}(i, \mathbf{w})$ генерируется улучшенная стратегия μ. Эта новая стратегия должна быть жадной по отношению к $\hat{J}^{\mu}(i, \mathbf{w})$ для всех i.

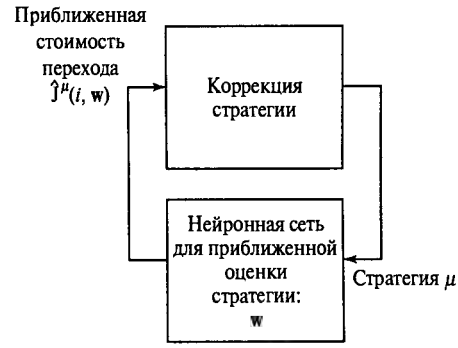


Рис. 12.7. Упрощенная блочная диаграмма приближенного алгоритма итерации по стратегиям

Для того чтобы приближенный алгоритм итерации по стратегиям давал удовлетворительные результаты, важно тщательно выбирать стратегию, используемую для инициализации алгоритма. При этом необходимо использовать эвристики. В качестве альтернативы можно начинать с произвольного вектора $\mathbf{w}$ и использовать его для вывода жадной стратегии. Полученная стратегия затем будет применяться в качестве исходной.

Теперь предположим, что в дополнение к знанию вероятностей переходов и значений стоимости в нашем распоряжении имеется следующая информация.

- Стационарная стратегия μ, принимаемая в качестве исходной.
- Множество состояний $\mathbf{X}$, представительных для данной внешней среды.
- Множество примеров $M(i)$ функций *стоимости перехода* $J^{\mu}(i)$ для всех состояний $i \in \mathbf{X}$. Каждый такой пример обозначим как $k(i, m)$, где $m = 1, 2, \ldots, M(i)$.

Пусть $\hat{J}^{\mu}(i, \mathbf{w})$ — приближенное представление функции стоимости перехода $J^{\mu}(i)$. Эта аппроксимация выполняется нейронной сетью (например, многослойным персептроном, обучаемым с помощью алгоритма обратного распространения). Вектор параметров $\mathbf{w}$ этой нейронной сети определяется с использованием метода наименьших квадратов, т.е. путем минимизации следующей функции стоимости:

$$\mathbf{E}(\mathbf{w}) = \sum_{i \in \mathbf{X}} \sum_{m=1}^{M(i)} (k(i, m) - \hat{J}^{\mu}(i, \mathbf{w}))^2. \tag{12.32}$$

Определив оптимальный вектор весов и, следовательно, приближенную функцию стоимости перехода $\hat{J}^{\mu}(i, \mathbf{w})$, приближенные Q-факторы можно вычислить по следующей формуле:

$$Q(i, a, \mathbf{w}) = \sum_{j \in \mathbf{X}} p_{ij}(a)(g(i, a, j) + \gamma \hat{J}^{\mu}(j, \mathbf{w})), \tag{12.33}$$

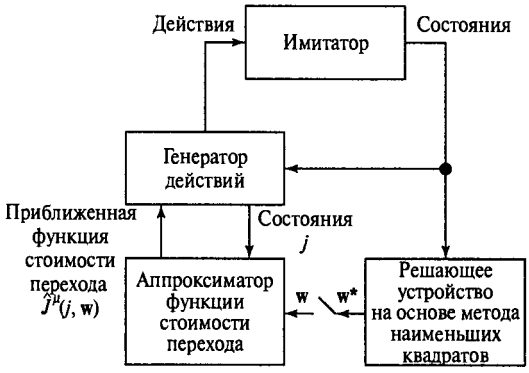


Рис. 12.8. Детальная схема приближенного алгоритма итерации по стратегиям

где $p_{ij}(a)$ — вероятности перехода из состояния i в состояние j при (известном) действии a; $g(i, a, j)$ — наблюдаемые (известные) затраты; γ — дисконтный множитель. Итерация завершается вычислением улучшенной стратегии с использованием полученного приближенного Q-фактора (см. (12.28))

$$\mu(i) = \arg\min_{a \in \mathbf{A}_i} Q(i, a, \mathbf{w}). \tag{12.34}$$

Важно отметить, что выражения (12.33) и (12.34) используются модельной сетью для генерации действий, выполняемых в состояниях, *реально посещаемых* при моделировании, а не во состояниях. Поэтому данные два выражения не подвержены проклятию размерности.

Блочная диаграмма на рис. 12.8 представляет собой более детализированную иллюстрацию приближенного алгоритма итерации по стратегиям. Эта диаграмма состоит из четырех взаимосвязанных модулей [126].

1. *Имитатор* (simulator). Использует данные вероятности переходов и стоимости для одного шага (мгновенные затраты) для построения суррогатной модели внешней среды. Имитатор генерирует два типа данных: состояния в ответ на действия (имитация внешней среды) и реализации функции стоимости перехода для данной стратегии.
2. *Генератор действий* (action generator). Генерирует улучшенную стратегию (т.е. последовательность действий) в соответствии с (12.34).
3. *Аппроксиматор функции стоимости перехода* (cost-to-go approximator). Генерирует приближенную функцию стоимости перехода $\hat{J}^{\mu}(i, \mathbf{w})$ для состояния i и вектора параметров $\mathbf{w}$ выражений (12.33) и (12.34).
4. *Решающее устройство на основе метода наименьших квадратов* (least-square solver). Ему поподаются реализации функции стоимости перехода $J^{\mu}(i)$, сгене-

рированные имитатором для стратегии μ и состояния i, для которых он вычисляет оптимальный вектор параметров $\mathbf{w}$, минимизирующий функцию стоимости (12.32). Связь, ведущая от решающего устройства к имитатору функции стоимости перехода, отключается только после получения полной оценки стратегии и нахождения оптимального вектора параметров $\mathbf{w}^*$. В этой точке аппроксимация $\hat{J}^{\mu}(i, \mathbf{w})$ заменяется функцией $\hat{J}^{\mu}(i, \mathbf{w}^*)$.

В табл. 12.3 приведен приближенный алгоритм итерации по стратегиям в сжатом виде.

ТАБЛИЦА 12.3. Приближенный алгоритм итерации по стратегиям

Известные параметры

Вероятности перехода $p_{ij}(a)$ и затраты $g(i, a, j)$

Вычисления

1. Выбираем некоторую стационарную стратегию μ в качестве исходной
2. Используя множество примеров $\{k(i,m)\}_{m=1}^{M(i)}$ функции стоимости перехода $J^{\mu}(i)$, генерируемые имитатором, определяем вектор параметров $\mathbf{w}$ нейронной сети с помощью решающего устройства, работающего на основе метода наименьших квадратов:
$$\mathbf{w}^* = \min_{\mathbf{w}} \mathbf{E}(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i \in \mathbf{X}} \sum_{m=1}^{M(i)} (k(i,m) - \hat{J}^{\mu}(i, \mathbf{w}))^2$$
3. Для вектора параметров $\mathbf{w}$, вычисленного на шаге 2, вычисляем приближенную функцию стоимости перехода $\hat{J}^{\mu}(i, \mathbf{w})$ для посещенных состояний. Определяем приближенные Q-факторы
$$Q(i, a, \mathbf{w}) = \sum_{j \in \mathbf{X}} p_{ij}(a)(g(i, a, j) + \gamma \hat{J}^{\mu}(j, \mathbf{w}))$$
4. Определяем улучшенную стратегию: $\mu(i) = \arg\min_{a \in \mathbf{A}_i} Q(i, a, \mathbf{w})$
5. Повторяем шаги 2–4

Естественно, работа этого алгоритма подвержена ошибкам ввиду неизбежного несовершенства архитектуры имитатора и решающего устройства. Для нейронной сети, используемой для вычисления аппроксимации искомой функции стоимости перехода по методу наименьших квадратов, может не хватить вычислительных мощностей. Это первый источник ошибок. Оптимизация аппроксиматора нейронной сети и, таким образом, настройки вектора параметров $\mathbf{w}$ основана на желаемом отклике, предоставляемом имитатором. Это второй источник ошибок. В предположении, что все оценки стратегии и все модификации стратегий выполняются с определенными допусками ε и δ, в [126] было показано, что приближенный алгоритм итерации по стратегиям дает на выходе такие стратегии, производительность которых отличается от оптимальной на величину, стремящуюся к нулю при уменьшении ε и δ.

Другими словами, приближенный алгоритм итерации по стратегиям не гарантирует существенного улучшения производительности. Согласно [126], приближенный алгоритм итерации по стратегиям вначале монотонно и довольно быстро сходится, однако устойчивая осцилляция стратегии, имеющая случайную природу, может привести к снижению скорости сходимости. Эта осцилляция появляется после того, как приближенная функция стоимости перехода попадает в окрестность $O((\delta+2\gamma\varepsilon)/(1-\gamma)^2)$ оптимального значения J^*, где γ — дисконтный параметр. Очевидно, что существует некоторая фундаментальная структура, общая для всех вариантов приближенного алгоритма итерации по стратегиям, которая вызывает такую осцилляцию.

12.8. Q-обучение

Поведенческой задачей системы обучения с подкреплением (см. рис. 12.1) является поиск оптимальной стратегии (т.е. имеющей минимальную стоимость) в результате попытки выполнения возможных последовательностей действий и оценки соответствующих стоимостей и переходов. В этом контексте возникает следующий вопрос: существует ли процедура обучения оптимальной стратегии в реальном времени, основанная на опыте, накопленном только на примерах следующего вида:

$$s_n = (i_n, a_n, j_n, g_n), \tag{12.35}$$

где n — дискретное время, а s_n состоит из четверки величин: действия a_n в состоянии i_n, которое приводит к переходу в состояние $j_n = i_{n+1}$ при затратах $g_n = g(i_n, a_n, j_n)$. Ответ на этот фундаментальный вопрос является утвердительным, и его описывает стохастический метод, получивший название *Q-обучения*[4] (Q-learning) [1115]. Q-обучение представляет собой пошаговую процедуру динамического программирования, определяющую оптимальную стратегию. Эта процедура идеально подходит для решения Марковских задач принятия решений при отсутствии явных знаний о вероятностях переходов. Однако успешное использование Q-обучения основано на предположении, что состояния внешней среды являются *вполне наблюдаемыми* (fully observable), а это, в свою очередь, значит, что среда является вполне наблюдаемой цепью Маркова.

---

[4] На с. 96 кандидатской диссертации [1115] Q-обучению описывается так.

"В приложении 1 представлено доказательство того, что этот метод работает для конечных Марковских процессов принятия решений. При этом также показано, что данный метод обучения будет быстро сходиться к оптимальной функции. Хотя эта идея является достаточно простой, насколько я знаю, раньше ей не уделялось внимание. Следует сказать, что конечные Марковские процессы принятия решений и стохастическое динамическое программирование на протяжении более 30 лет тщательно изучались на предмет использования их в различных областях, но, к сожалению, никто не догадался применить метод Монте-Карло".

В сноске к этому замечанию в [99] отмечалось, что, несмотря на то что идея присваивания значений парам состояние–действие на основе метода динамического программирования высказана в [253], в ней не был сформулирован алгоритм, подобный Q-обучению, для оценки этих значений. Это было сделано только в диссертации [1115].

В разделе 12.4 было введено определение Q-фактора $Q(i,a)$ для пары состояние–действие (i,a) с помощью выражения (12.23) и уравнение оптимальности Беллмана, описываемое выражением (12.22). Объединяя эти два выражения и используя определение *мгновенных ожидаемых затрат* (immediate expected cost) (12.20), получим:

$$Q^*(i,a) = \sum_{j=1}^{N} p_{ij}(a)\left(g(i,a,j) + \gamma \min_{b \in \mathbf{A}_j} Q^*(j,b)\right) \quad \text{для всех } (i,a). \tag{12.36}$$

Это выражение можно рассматривать как двухшаговую версию уравнения оптимальности Беллмана. Решения линейной системы уравнений (12.36) определяют оптимальные Q-факторы $Q^*(i,a)$, единственные для пары состояние–действие (i,a). Для решения этой системы уравнений можно использовать алгоритм итерации по значениям, сформулированный в терминах Q-факторов. Таким образом, для одной итерации этого алгоритма имеем:

$$Q(i,a) := \sum_{j=1}^{N} p_{ij}(a)\left(g(i,a,j) + \gamma \min_{b \in \mathbf{A}_j} Q(j,b)\right) \quad \text{для всех } (i,a).$$

Предполагая малость шага, это выражение можно переписать так:

$$Q(i,a) := (1-\eta)Q(i,a) + \eta \sum_{j=1}^{N} p_{ij}(a)\left(g(i,a,j) + \gamma \min_{b \in \mathbf{A}_j} Q(j,b)\right) \tag{12.37}$$

для всех (i,a),

где η — *малый параметр скорости обучения*, который находится в диапазоне $0 < \eta < 1$.

Шаг алгоритма итерации по значениям (12.37) требует знания вероятностей переходов. Однако эти априорные знания не понадобятся, если сформулировать *стохастическую* версию выражения (12.37). В частности, усреднение по множеству состояний, осуществляемое в (12.37), заменяется единственным примером, что приводит к следующей модификации выражения коррекции Q-фактора:

$$Q_{n+1}(i,a) = (1-\eta_n(i,a))Q_n(i,a) + \eta_n(i,a)\left[g(i,a,j) + \gamma J_n(j)\right]$$
$$\text{для } (i,a) = (i_n, a_n), \tag{12.38}$$

где

$$J_n(j) = \min_{b \in \mathbf{A}_j} Q_n(j,b), \tag{12.39}$$

j — последующее состояние; $\eta_n(i,a)$ — параметр скорости обучения на шаге n для пары состояние–действие (i,a). Уравнение коррекции (12.38) применяется к текущей паре состояние–действие (i_n,a_n), для которой, согласно (12.35), $j=j_n$. Для всех остальных допустимых пар состояние–действие Q-факторы остаются неизменными:

$$Q_{n+1}(i,a)=Q_n(i,a)(i,a)\neq(i_n,a_n). \tag{12.40}$$

Уравнения (12.38)–(12.40) составляют одну итерацию *алгоритма Q-обучения.*

Теорема о сходимости[5]

Предположим, что параметр скорости обучения $\eta_n(i,a)$ *удовлетворяет следующим условиям:*

$$\sum_{n=0}^{\infty}\eta_n(i,a)=\infty \text{ и } \sum_{n=0}^{\infty}\eta_n^2(i,a)<\infty \text{ для всех } (i,a). \tag{12.41}$$

Тогда последовательность Q-факторов $\{Q_n(i,a)\}$*, генерируемых алгоритмом Q-обучения, сходится с вероятностью 1 к оптимальному значению* $Q^*(i,a)$ *для всех пар состояние–действие* (i,a) *при стремлении количества итераций n к бесконечности (при этом предполагается, что все пары состояние–действие посещаются бесконечно часто).*

Примером параметра скорости обучения, зависящего от времени, который гарантирует сходимость данного алгоритма, может служить следующий:

$$\eta_n=\frac{\alpha}{\beta+n},\quad n=1,2,\ldots, \tag{12.42}$$

где α и β — положительные числа.

Подводя итог, можно сделать следующий вывод. Алгоритм Q-обучения является приближенной формой стратегии итерации по значениям. На каждой итерации алгоритм резервирует Q-фактор для *одной* пары состояние–действие, а именно для текущего наблюдаемого состояния и фактически выполняемого действия. Что более важно — этот алгоритм в пределе сходится к оптимальным Q-значениям без формирования явной модели соответствующих Марковских процессов принятия решений. Как только оптимальные значения Q-факторов становятся известны, оптимальная стратегия с помощью выражения (12.30) может быть определена без особо трудоемких вычислений.

---

[5] Схема доказательства теоремы о сходимости Q-обучения была представлена в [1115]. Само детальное доказательство приводится в [1116]. Более общие результаты относительно сходимости Q-обучения были описаны в [126], [1059].

Сходимость Q-обучения к оптимальной стратегии предполагает представление Q-факторов $Q_n(i,a)$ в виде *справочных таблиц* (look-up table). Этот способ представления является достаточно простым и вычислительно эффективным. Однако если входное пространство, состоящее из пар состояние–действие, является достаточно большим или если входные переменные непрерывны, использование справочных таблиц может быть недопустимо затратным ввиду требования громадных объемов памяти. В такой ситуации для аппроксимации функций можно использовать нейронные сети.

Приближенное Q-обучение

Равенства (12.38) и (12.39) определяют формулы коррекции Q-фактора для текущей пары состояние–действие (i_n, a_n). Эта пара равенств может быть переписана в эквивалентном виде:

$$Q_{n+1}(i_n,a_n) = Q_n(i_n,a_n) + \\ +\eta_n(i_n,a_n)\left[g(i_n,a_n,j_n) + \gamma \min_{b\in \mathbf{A}_{jn}} Q_n(j_n,b) - Q_n(i_n,a_n)\right]. \tag{12.43}$$

Рассматривая выражение в квадратных скобках в правой части (12.43) как сигнал ошибки в формуле коррекции текущего Q-фактора $Q_n(i_n,a_n)$, можно определить целевой (желаемый) Q-фактор в момент времени n как

$$Q_n^{\text{целевой}}(i_n,a_n) = g(i_n,a_n,j_n) + \gamma \min_{b\in \mathbf{A}_{jn}} Q_n(j_n,b), \tag{12.44}$$

где $j_n = i_{n+1}$ — последующее состояние. Уравнение (12.44) показывает, что последующее состояние j_n играет важнейшую роль в определении целевого Q-фактора. Используя определение целевого Q-фактора, можно переформулировать алгоритм Q-обучения в следующем виде:

$$Q_{n+1}(i,a) = Q_n(i,a) + \Delta Q_n(i,a), \tag{12.45}$$

где пошаговое изменение текущего Q-множителя определяется формулой

$$\Delta Q_n(i,a) = \begin{cases} \eta_n(Q_n^{\text{целевой}}(i,a) - Q_n(i,a)) & \text{при } (i,a) = (i_n,a_n), \\ 0 \text{ — в противном случае.} \end{cases} \tag{12.46}$$

По определению "оптимальное" действие a_n в текущем состоянии i_n является конкретным действием в том состоянии, для которого Q-фактор в момент времени n минимален. Исходя из этого, при данных Q-факторах $Q_n(i_n,a)$ для допустимых действий $a \in \mathbf{A}_{in}$ в состоянии i_n оптимальное действие, подставляемое в выражение (12.44), представляется следующей формулой:

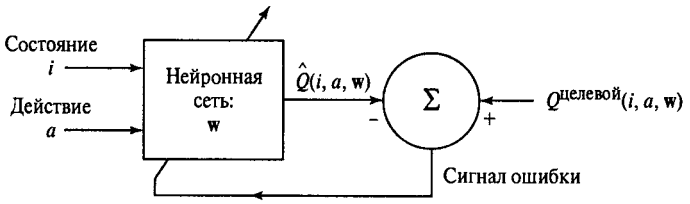


Рис. 12.9. Архитектура нейронной сети, аппроксимирующей целевой Q-фактор $Q^{\text{целевой}}(i, a, \mathbf{w})$

$$Q_n = \min_{a \in \mathbf{A}_{i_n}} Q_n(i_n, a). \tag{12.47}$$

Обозначим символом $\hat{Q}_n(i_n, a_n, \mathbf{w})$ аппроксимацию Q-фактора $Q_n(i_n, a_n)$, вычисленную нейронной сетью (например, многослойным персептроном, обучаемым согласно алгоритму обратного распространения). Текущая пара состояние–действие (i_n, a_n) является входом нейронной сети с вектором параметров $\mathbf{w}$, генерирующей выходной сигнал $\hat{Q}_n(i_n, a_n, \mathbf{w})$ (рис. 12.9). На каждой итерации этого алгоритма вектор весов $\mathbf{w}$ медленно изменяется таким образом, чтобы как можно лучше приблизить выход $\hat{Q}_n(i_n, a_n, \mathbf{w})$ к целевому значению $Q_n^{\text{целевой}}(i_n, a_n)$. Однако как только вектор $\mathbf{w}$ изменяется, это неявно влияет на само целевое значение. При этом подразумевается, что модифицируемое значение – $Q_n^{\text{целевой}}(i_n, a_n, \mathbf{w})$. Таким образом, нет никаких гарантий, что расстояние между двумя Q-факторами будет уменьшаться на каждой итерации. Это является еще одной причиной, почему алгоритм приближенного Q-обучения потенциально может расходиться. Если алгоритм не расходится, вектор весов $\mathbf{w}$ обеспечивает средства хранения аппроксимированных Q-факторов в обучаемой нейронной сети, так как ее выход $\hat{Q}_n(i_n, a_n, \mathbf{w})$ представляет собой отклик на вход (i_n, a_n).

В табл. 12.4 в сжатом виде приведен алгоритм приближенного Q-обучения.

Исследование

В алгоритме итерации по стратегиям необходимо исследовать все потенциально важные части пространства состояний. При использовании Q-обучения вводится еще одно дополнительное требование: необходимо также исследовать все потенциально прибыльные действия. В частности, чтобы удовлетворить условиям теоремы сходимости, все допустимые пары состояние–действие должны исследоваться достаточно часто. Для жадной стратегии μ исследуются только некоторые пары состояние–действие $(i, \mu(i))$. К сожалению, нет никакой гарантии, что даже в исследовании всего пространства пар состояние–действие будут опробованы все выгодные действия.

Необходимо предложить стратегию, расширяющую Q-обучение за счет введения компромисса между двумя антагонистическими целями [1053].

- *Исследованием* (exploration), гарантирующим, что все допустимые пары состояние–действие будут рассматриваться достаточно часто, чтобы удовлетворить теореме о сходимости Q-обучения.
- *Эксплуатацией* (exploitation), которая стремится минимизировать функцию стоимости перехода, следуя жадной стратегии.

Одним из способов достижения компромисса является следование *смешанной нестационарной стратегии* (mixed nonstationary policy), которая переключается между вспомогательным Марковским процессом и исходным Марковским процессом, управляемым стационарной жадной стратегией, определяемой Q-обучением [235]. Вспомогательный процесс имеет следующую интерпретацию. Вероятности перехода между возможными состояниями определяются вероятностями перехода исходного управляемого процесса с добавлением условия, что соответствующие действия равномерно распределены. Эта смешанная стратегия начинается с произвольного состояния вспомогательного процесса и выбирает соответствующие действия, после этого переключается на исходный управляемый процесс и т.д. (рис. 12.10). Время, затраченное на работу вспомогательного процесса, занимает фиксированное число шагов L, определяемое как удвоенное максимальное время, требуемое для посещения всех состояний вспомогательного процесса. Время, затраченное на работу основного управляемого процесса, с каждым следующим переключением постепенно увеличивается. Пусть n_k — время, в которое мы переключаемся со вспомогательного процесса на основной, а m_k — время переключения назад во вспомогательный процесс, где величины определяются следующим образом:

$$n_k = m_{k-1} + L, k = 1, 2, \ldots \text{ и } m_0 = 1,$$
$$m_k = n_k + kL, k = 1, 2, \ldots.$$

Вспомогательный процесс строится таким образом, чтобы при стремлении k к бесконечности с вероятностью 1 существовало бесконечное количество посещений всех состояний, что будет гарантировать сходимость к оптимальным Q-факторам. Более того, при $k \to \infty$ время, отданное смешанной стратегией работе вспомогательного процесса, становится асимптотически малой долей времени, отданного работе основного управляемого процесса. Это, в свою очередь, означает, что смешанная стратегия асимптотически сходится к некоторой жадной стратегии. Тогда, в предположении сходимости Q-факторов к своим оптимальным значениям, жадная стратегия действительно становится оптимальной, при условии, что эта стратегия становится жадной достаточно медленно.

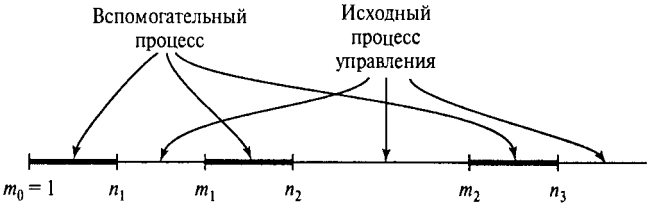


Рис. 12.10. Временные интервалы, относящиеся к вспомогательному и основному управляемому процессам

ТАБЛИЦА 12.4. Алгоритм приближенного Q-обучения

1. Начинаем с произвольного вектора весов $\mathbf{w}_0$, на основе которого получаем Q-фактор $Q(i_0, a_0, \mathbf{w}_0)$. Вектор весов $\mathbf{w}$ относится к нейронной сети, используемой для осуществления аппроксимации
2. Для итераций $n = 1, 2, \ldots$ выполняем следующее

2а. Для данного вектора весов $\mathbf{w}$ нейронной сети определяем оптимальное действие:

$$a_n = \min_{a \in \mathbf{A}_{i_n}} Q_n(i_n, a, \mathbf{w})$$

2б. Определяем целевой Q-фактор:

$$Q_n^{\text{целевой}}(i_n, a_n, \mathbf{w}) = g(i_n, a_n, j_n) + \gamma \min_{b \in \mathbf{A}_{j_n}} Q_n(j_n, b, \mathbf{w})$$

2в. Корректируем Q-фактор

$$Q_{n+1}(i_n, a_n, \mathbf{w}) = Q_n(i_n, a_n, j_n) + \Delta Q_n(i_n, a_n, \mathbf{w}),$$

где

$$\Delta Q_n(i_n, a_n, \mathbf{w}) = \begin{cases} \eta_n(i_n, a_n)(Q_n^{\text{целевой}}(i_n, a_n, \mathbf{w}) - Q_n(i_n, a_n, \mathbf{w})) \\ \qquad\qquad \text{при } (i, a) = (i_n, a_n), \\ 0 \text{ — в противном случае} \end{cases}$$

2г. Применяем (i_n, a_n) в качестве входа нейронной сети, генерирующей выходной сигнал $\hat{Q}_n(i_n, a_n, \mathbf{w})$, являющийся аппроксимацией целевого Q-фактора $Q_n^{\text{целевой}}(i_n, a_n, \mathbf{w})$. Слегка изменяем вектор весов $\mathbf{w}$ так, чтобы приблизить аппроксимацию $\hat{Q}_n(i_n, a_n, \mathbf{w})$ к целевому значению $Q_n^{\text{целевой}}(i_n, a_n, \mathbf{w})$

2д. Возвращаемся к шагу 2а и повторяем вычисления

12.9. Компьютерный эксперимент

В этом компьютерном эксперименте вновь вернемся к задаче дилижанса, рассмотренной в примере 12.1. На этот раз для решения задачи будем использовать приближенное Q-обучение. Для реализации этого алгоритма существуют два основных подхода: в первом из них для представления Q-значений используется справочная таблица, а во втором — нейронная сеть.

На рис. 12.11 показаны истории обучения следующих Q-факторов: $Q(A, \text{вверх})$, $Q(C, \text{прямо})$, $Q(E, \text{прямо})$ и $Q(I, \text{вверх})$. Пунктирными линиями на рис. 12.11 представлены желаемые Q-значения. Каждая попытка представляет собой полный маршрут от штата i до штата J. Начальное состояние для каждой попытки выбирается случайным образом. Параметр скорости обучения $\eta_n(i, a)$ определяется соотношением

$$\eta_n(i, a) = \frac{\alpha v_n(i, a)}{K + v_n(i, a)},$$

где $v_n(i, a)$ — количество посещений состояния (i, a) до текущего момента n; $\alpha = 1,6$ и $K = 600$. После совершения 1000 попыток был определен оптимальный маршрут:

$$A \rightarrow D \rightarrow F \rightarrow I \rightarrow J.$$

Это один из оптимальных маршрутов с общими затратами, равными 11. Это же значение было получено в примере 12.1.

На рис. 12.12 показаны соответствующие результаты, полученные с помощью многослойного персептрона с двумя входными узлами, десятью скрытыми и одним выходным нейроном. Один из входных узлов представляет состояние, а второй — действие, предпринимаемое для перехода из одного состояния в другое. Выход этого многослойного персептрона представляет Q-значение, вычисленное нейронной сетью. Эта нейронная сеть обучалась с использованием стандартного алгоритма обратного распространения. Целевое Q-значение в момент времени n вычислялось по формуле (12.44). Параметр скорости обучения принимал фиксированное значение 0,012, при этом не использовались моменты. Эта сеть обучалась на 10 000 примерах для каждой пары состояние–действие. На рис. 12.12 показана история обучения следующих Q-значений: $Q(A, \text{вверх})$, $Q(C, \text{прямо})$, $Q(E, \text{прямо})$ и $Q(I, \text{вверх})$. Оптимальный маршрут, найденный сетью, выглядит следующим образом:

$$A \rightarrow D \rightarrow E \rightarrow H \rightarrow J.$$

Это также один из оптимальных маршрутов, имеющих общие затраты, равные 11.

Вычислительные требования для применения этих двух подходов приведены ниже.

1. Нейронная сеть.
   Количество входов — 2.
   Количество скрытых нейронов — 10.
   Количество выходных нейронов — 1.
   Общее количество синаптических весов и порогов — $2 \times 10 + 10 + 10 \times 1 + 1 = 41$.

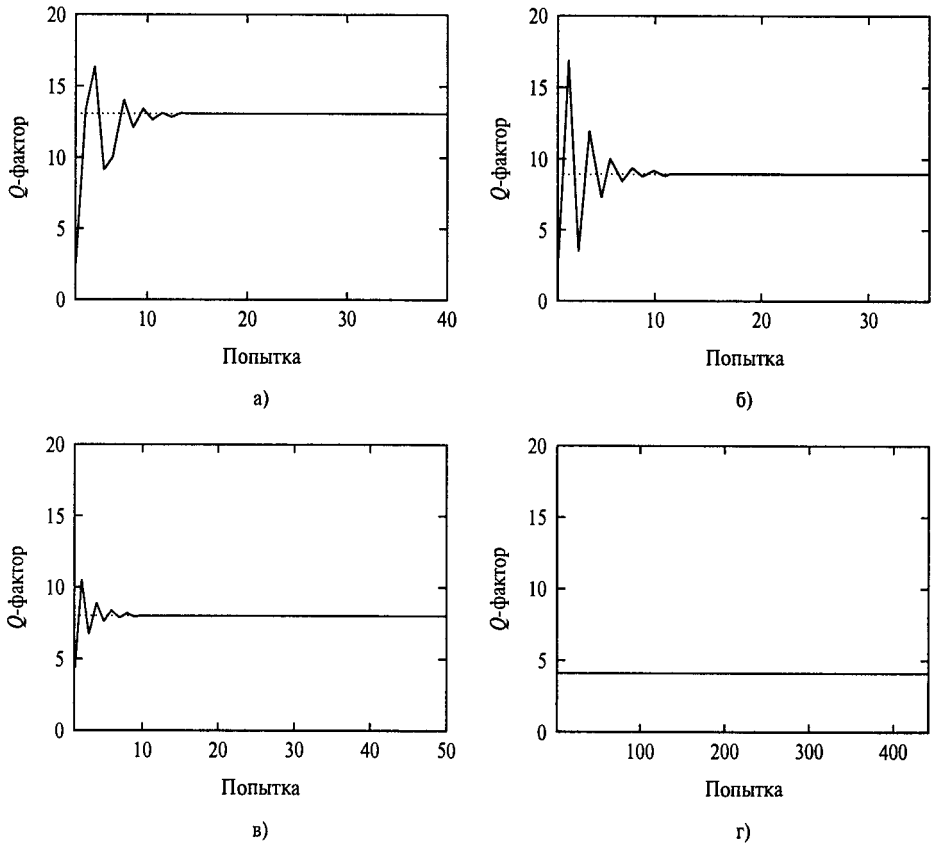


Рис. 12.11. Кривые обучения для задачи дилижанса при использовании справочных таблиц: кривая обучения для $Q(A, \text{вверх})$ (а); для $Q(C, \text{прямо})$ (б); для $Q(E, \text{прямо})$ (в) и для $Q(I, \text{вверх})$ (г)

2. Справочная таблица.
   Количество состояний — 10.
   Количество действий — 2 или 3.
   Размер таблицы — 21.

В этом эксперименте количество возможных состояний мало, в результате чего для хранения справочной таблицы требуется меньше памяти, чем для нейронной сети. Однако если количество состояний достаточно велико, нейронная сеть обладает преимуществами по сравнению со справочной таблицей с точки зрения требуемых вычислительных мощностей.

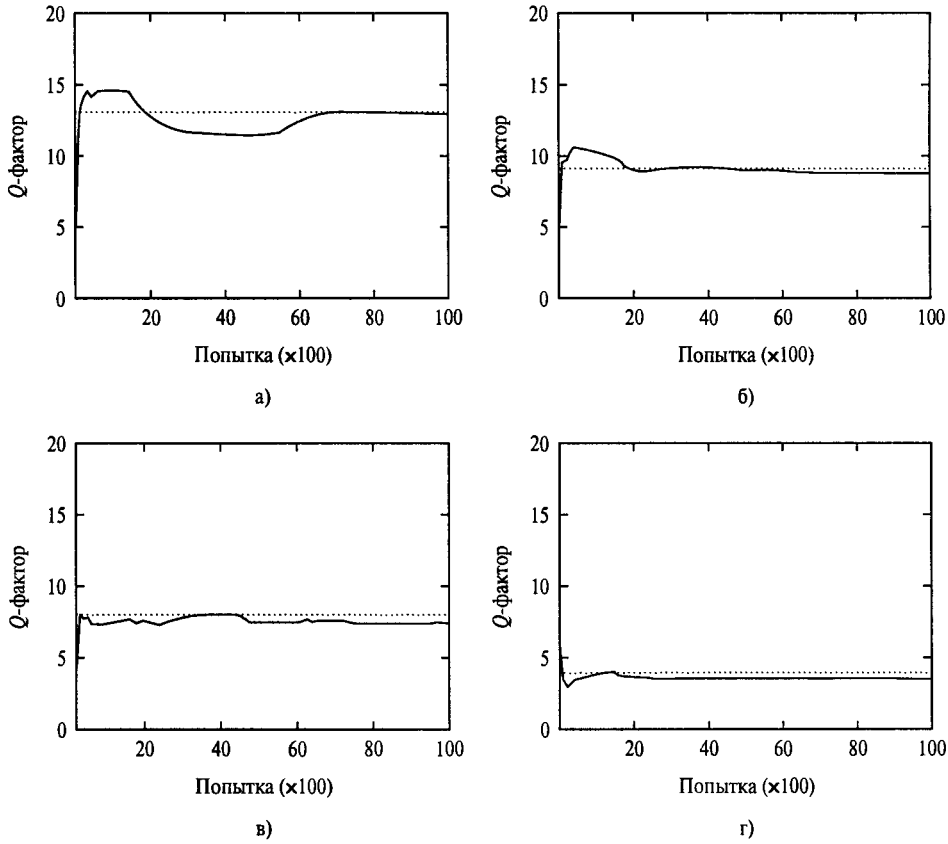


Рис. 12.12. Кривые обучения для задачи дилижанса при использовании нейронной сети. Кривая обучения для $Q(A, \text{вверх})$ (а); для $Q(C, \text{прямо})$ (б); для $Q(E, \text{прямо})$ (в) и для $Q(I, \text{вверх})$ (г)

12.10. Резюме и обсуждение

Нейродинамическое программирование, соединяющее в себе математический формализм классического динамического программирования и способность нейронной сети к обучению, обеспечивает мощный подход к решению поведенческих задач, требующих планирования. В этом современном подходе к обучению с подкреплением система обучается двум вещам: на основании наблюдений за собственным поведением принимать грамотные решения и улучшать их действия с помощью механизма подкрепления. Лежащий в его основе процесс принятия решений соответствует Марковской модели.

В этой главе описаны две процедуры нейродинамического программирования.

1. *Приближенная итерация по стратегиям.* Итерации по стратегиям соответствует два основных шага.
   - Шаг вычисления стратегии, на котором вычисляется функция стоимости перехода для некоторой текущей стратегии.
   - Шаг улучшения стратегии, в котором текущая стратегия корректируется для того, чтобы стать жадной по отношению к текущей функции стоимости перехода.

   В приближенной итерации по стратегиям при вычислении стратегии объединяются моделирование и аппроксимация функций. Для создания Марковской модели системы требуется знание вероятностей перехода между состояниями. Для получения аппроксимации функции можно использовать нейронную сеть (например, многослойный персептрон, обучаемый согласно алгоритму обратного распространения, или машина опорных векторов), для которой эта задача подходит с точки зрения свойства универсального аппроксиматора.
2. *Приближенное Q-обучение.* При итерации по значениям, альтернативной итерации по стратегиям, Марковская задача принятия решений решается с использованием последовательной процедуры аппроксимации, сходящейся к оптимальной стратегии. Q-обучение представляет собой асинхронную форму метода итерации по значениям, сформулированную для избежания необходимости явных знаний о вероятностях переходов между состояниями. Этот метод обладает следующими привлекательными особенностями.
   - Q-обучение сходится к оптимальным Q-значениям с вероятностью 1 при условии, что все пары значение–действие будут посещаться бесконечно часто, а параметры скорости обучения удовлетворяют условиям (12.41).
   - Q-обучение напрямую корректирует оценки Q-факторов, связанных с оптимальной стратегией, и, таким образом, устраняет потребность в многочисленных шагах оценки, которые присутствуют в методе итерации по стратегиям.

В приближенном Q-обучении для аппроксимации оценок Q-факторов используется нейронная сеть. Это помогает избежать больших затрат памяти в задачах с большим количеством состояний. Короче говоря, приближенное Q-обучение является алгоритмом решения Марковской задачи принятия решений, основанным на моделировании. Этот алгоритм применяется, когда модель системы недоступна, а требования к экономии памяти стоят на первом месте. Естественно, его можно применить даже в случае доступности модели системы. В этом случае он становится альтернативой приближенному алгоритму итерации по стратегиям.

Методы нейродинамического программирования особенно эффективны при решении масштабных задач, когда на первом месте стоит планирование. Традиционные подходы динамического программирования сложно применить к задачам такого типа, так как при этом необходимо исследовать пространство состояний слишком большой размерности. Нейродинамическое программирование успешно применялось на практике для решения сложных задач в различных прикладных областях, в том числе в игре в нарды (backgammon) [1046], [1048], в комбинаторной оптимизации [126], *диспетчеризации* (elevator dispatching) [233] и при *динамическом распределении каналов* (dynamical channel allocation) [783], [784], [996]. Ниже приложение к игре в нарды будет описано более детально.

Программа-игрок в триктак, основанная на нейронных сетях, впервые была описана в [1048] и впоследствии была улучшена в [1046]. Этот успех для частного случая послужил толчком к дальнейшим исследованиям в области нейродинамического программирования. Триктрак (нарды) — это древняя настольная игра, в которой принимают участие два игрока. Она может быть представлена как одномерный маршрут, который шашки игроков проходят в противоположных направлениях. Игроки поочередно бросают пару костей и соответствующим образом передвигают свои шашки по маршруту доски. Допустимые перемещения зависят от выпавшего на костях количества очков и конфигурации доски. Каждый из игроков должен переместить все свои шашки из конца маршрута, принадлежащего противнику, в свой конец маршрута. Первый, кто выполнит эту задачу, считается победителем. Эту игру можно промоделировать как Марковский процесс принятия решений, в котором состояния определяются описанием конфигурации доски, выпавшими на костях очками и идентификатором игрока, совершающего ход. Первая версия нейронардов, описанная в [1048], использует обучение с учителем. Она способна обучаться на промежуточном уровне при наличии только приближенного описания состояния. Пожалуй, самым интересным открытием, описанным в этой работе, было хорошее поведение при масштабировании: при увеличении размера нейронной сети и количества примеров обучения наблюдалось соответствующее увеличение производительности. Нейронную сеть, используемую при обучении, представлял многослойный персептрон (MLP), обучаемый согласно алгоритму обратного распространения. Наилучшей эффективности удалось добиться при использовании MLP с 40 скрытыми нейронами. При этом обучение проводилось в ходе более чем 200000 игр. В последующих исследованиях [1046] для обучения нейронной сети использовалась одна из форм итерации по стратегиям, названная *оптимистической* $TD(\lambda)$. Аббревиатура TD означает temporal difference learning — *обучение на временных разностях* [1032]. Оптимистическая стратегия $TD(\lambda)$ — это метод аппроксимации функции стоимости перехода J^{μ}, основанный на моделировании, в котором стратегия μ заменяется новой стратегией, являющейся жадной по отношению к аппроксимации J^{μ} при каждом переходе состояний [126]. Компьютерную программу, основанную на этом методе нейродинамического программирования, часто называют

TD-шулером. В представление входного сигнала для нейронной сети Тезауро (Tesauro) добавил искусственно созданные функции состояний (т.е. признаки), что позволило этой программе играть на уровне мастера, приближающегося к уровню чемпиона мира. Убедительным основанием для этой оценки послужил ряд тестов, в которых эта программа играла против нескольких гроссмейстеров мирового класса [1045].

Задачи

Критерий оптимальности Беллмана

12.1. При стремлении дисконтного множителя γ к нулю вычисление функции стоимости перехода (12.22) замедляется. Почему это происходит? Обоснуйте свой ответ.

12.2. В этой задаче будет представлено еще одно доказательство уравнения оптимальности Беллмана (12.22) [905].

а) Пусть π — произвольная стратегия. Предположим, что π выбирает действие a в момент времени 0 с вероятностью p_a, где $a \in \mathbf{A}_i$. Тогда

$$J^{\pi}(i) = \sum_{a \in \mathbf{A}_i} p_a \left(c(i,a) + \sum_{j=1}^{N} p_{ij}(a) W^{\pi}(j) \right),$$

где $W^{\pi}(j)$ представляет собой функцию ожидаемой стоимости перехода с шага 1 вперед при использовании стратегии π и состояния j. Покажите, что

$$J^{\pi}(i) \geq \min_{a \in \mathbf{A}_i} \left(c(i,a) + \gamma \sum_{j=1}^{N} p_{ij}(a) W^{\pi}(j) \right)$$

где

$$W^{\pi}(j) \geq \gamma J(i).$$

б) Пусть π — стратегия, которая выбирает действие a_0 в момент времени 0 и, если следующим состоянием будет j, рассматривает процесс, начинающийся в состоянии j и соответствующий стратегии π_j, такой, что

$$J^{\pi_i}(j) \leq J(j) + \varepsilon,$$

где ϵ — малое положительное число. Покажите, что при этих условиях

$$J^{\pi}(i) \geq \min_{a \in \mathbf{A}_i} \left(c(i,a) + \gamma \sum_{j=1}^{N} p_{ij}(a) J(j) \right) + \gamma\epsilon.$$

в) Используя результаты, полученные в пп. (а) и (б), получите равенство (12.22).

12.3. Уравнение (12.22) представляет собой систему N линейных уравнений, по одному для каждого состояния. Пусть

$$\mathbf{J}^{\mu} = [J^{\mu}(1), J^{\mu}(2), \ldots, J^{\mu}(N)]^T,$$
$$\mathbf{c}(\mu) = [c(1,\mu), c(2,\mu), \ldots, c(N,\mu)]^T,$$
$$\mathbf{P}(\mu) = \begin{bmatrix} p_{11}(\mu) & p_{12}(\mu) & \ldots p_{1N}(\mu) \\ p_{21}(\mu) & p_{22}(\mu) & \ldots p_{2N}(\mu) \\ \ldots & \ldots & \ldots\ldots \\ p_{N1}(\mu) & p_{N2}(\mu) & \ldots p_{NN}(\mu) \end{bmatrix}.$$

Покажите, что уравнение (12.22) можно переписать в эквивалентной матричной форме:

$$(\mathbf{I} - \gamma \mathbf{P}(\mu))\mathbf{J}^{\mu} = \mathbf{c}(\mu),$$

где $\mathbf{I}$ — единичная матрица. Прокомментируйте единственность вектора $\mathbf{J}^{\mu}$, представляющего функции стоимости перехода для N состояний.

12.4. В разделе 12.3 был выведен алгоритм динамического программирования для задачи с конечным горизонтом. В этой задаче мы снова выведем этот алгоритм для дисконтной задачи, в которой функция стоимости перехода выглядит следующим образом:

$$J^{\mu}(X_0) = \lim_{K \to \infty} \left[\sum_{n=0}^{K-1} \gamma^n g(X_n, \mu(X_n), X_{n+1}) \right].$$

В частности, покажите, что

$$J^{\mu}(X_0) = \min_{\mu} \underset{X_1}{E}[g(X_0, \mu(X_0), X_1) + \gamma J_{K-1}(X_1)].$$

Итерация по стратегиям

12.5. В разделе 12.4 говорилось, что функция стоимости перехода удовлетворяет условию

$$J^{\mu_{n+1}} \leq J^{\mu_n}.$$

Обоснуйте это утверждение.

12.6. Прокомментируйте значимость выражения (12.25).

12.7. Используя *систему контроллера с модулем критики* (controller critic system), проиллюстрируйте взаимодействие коррекции и оценки стратегий в алгоритме итерации по стратегиям.

Итерация по значениям

12.8. Задача динамического программирования включает в себя N возможных состояний и M допустимых действий. Предполагая использование стационарной стратегии, покажите, что одна итерация алгоритма итерации по значениям требует порядка N^2M операций.

12.9. В табл. 12.2 представлен в сжатом виде алгоритм итерации по значениям, сформулированный в терминах функции стоимости перехода $J^{\mu}(i)$ для состояний $i \in \mathbf{X}$. Переформулируйте этот алгоритм в терминах Q-факторов $Q(i,a)$.

12.10. Алгоритм итерации по стратегиям всегда завершается за конечное число шагов, в то время как алгоритм итерации по значениям может потребовать бесконечного их числа. Прокомментируйте другие различия между этими двумя методами динамического программирования.

Q-обучение

12.11. Покажите, что

$$J^*(i) = \min_{a \in \mathbf{A}_i} Q(i,a).$$

12.12. Алгоритм Q-обучения иногда называют адаптивной формой стратегии итерации по значениям. Обоснуйте справедливость этого утверждения.

12.13. Постройте граф передачи сигнала для приближенного алгоритма Q-обучения (см. табл. 12.4).

12.14. Приближенный алгоритм Q-обучения (см. табл. 12.4) предполагает отсутствие знаний о вероятностях перехода между состояниями. Переформулируйте этот алгоритм в предположении доступности этих вероятностей.

Временная обработка с использованием сетей прямого распространения

13.1. Введение

Время является существенной составляющей процесса обучения. Оно может быть непрерывным и дискретным. Какую бы форму ни принимало время, оно является упорядоченной сущностью, лежащей в основе большинства задач распознавания образов и речи, обработки сигнала и управления движением. Включение параметра времени в работу нейронной сети позволяет учитывать статистические вариации в таких нестационарных процессах, как речевые сигналы, сигналы радара, сигналы для мониторинга работы двигателя автомобиля и колебания цен на фондовом рынке, а также многих других. Возникает вопрос: "Как встроить время в описание работы нейронной сети?" Существуют два варианта ответа на этот фундаментальный вопрос.

- Используя *неявное представление* (implicit representation). Время представляется эффектом, который оно оказывает на обработку сигнала, т.е. неявным образом[1]. Для примера предположим, что входной сигнал имеет *равномерное квантование* (uniformly sampled), а последовательность синаптических весов каждого из нейронов, соединенных с входным слоем сети, получена с помощью свертки другой последовательности входных примеров. В такой системе временная структура входного сигнала вложена в пространственную структуру сети.
- Используя *явное представление* (explicit representation). Время имеет собственное конкретное представление[2]. Например, система эхо-локации летучей мыши посылает короткий частотно модулированный сигнал, при этом устанавливая единый уровень интенсивности для каждого из частотных каналов на короткий период FM-

[1] Обсуждение роли времени в нейронной обработке содержится в классической работе [281].

[2] В [475] описана методика явиого представления времени в нейронной обработке. В частности, аналоговая информация представляется с использованием синхронизации потенциалов действия (action potential) по отношению к постоянной составляющей колебательной модели действий, чему имеется очевидное нейробиологические объяснение. Потенциалы действия описаны в главе 1.

развертки. Для того чтобы извлечь точную информацию о расстоянии до цели, проводятся многочисленные сравнения нескольких различных частот, кодированных массивом слуховых рецепторов [1030]. Когда эхо получается от объекта с неизвестной задержкой, отвечает тот нейрон (слуховой системы), который имеет соответствующую задержку в линии. Таким образом оценивается расстояние до объекта.

В этой главе мы будем работать с неявным представлением времени, при котором "статическая" нейронная сеть (например, многослойный персептрон) обладает свойством *динамичности*. Это, в свою очередь, приводит к тому, что за временную структуру информационных сигналов отвечает сама сеть.

Чтобы нейронная сеть была динамичной, она должна иметь *память* (memory). Как уже говорилось в главе 2, память можно разделить на *кратковременную* и *долговременную*. Долговременная память встраивается в нейронную сеть посредством обучения с учителем, при котором информативное содержание множества данных обучения сохраняется (частично или в полном объеме) в синаптических весах сети. Однако если рассматриваемая задача также имеет и размерность времени, для придания сети свойств динамичности требуется какая-либо форма кратковременной памяти. Одним из простейших способов встраивания кратковременной памяти в структуру нейронной сети является использование *временных задержек* (time delay), которые можно реализовать на синаптическом уровне сети или в слое входных нейронов. Использование задержек по времени в нейронных сетях имеет свое нейробиологическое объяснение – хорошо известно, что задержки повсеместно встречаются в мозге и играют важную роль в нейробиологической обработке информации [146], [148], [150], [738].

Структура главы

Материал, излагаемый в настоящей главе, разбит на три части. В первой части (разделы 13.2, 13.3) рассматриваются структуры и модели нейронных сетей. В разделе 13.2 представлено обсуждение структур памяти, а в разделе 13.3 — описание двух различных архитектур сети, предназначенных для временной обработки сигнала.

Во второй части главы (разделы 13.4–13.6) рассматривается один из классов нейронных сетей, называемых сетями прямого распространения с фокусированным временем запаздывания (focused time lagged feedforward network). В этом названии термин "фокусированный" указывает на тот факт, что кратковременная память целиком расположена на переднем плане (front end) сети. Компьютерное моделирование этой структуры описывается в разделе 13.6.

В третьей части главы (разделы 13.7–13.9) рассматриваются сети прямого распространения с распределенным временем запаздывания (distributed time lagged feedforward network), в которых задержки в линии рассредоточены по всей сети. В разделе 13.7 описываются пространственно-временные модели нейрона, а в разделе 13.8 рассматривается вышеназванный класс нейронных сетей. В разделе 13.9 описывает-

ся "временной" алгоритм обратного распространения для обучения с учителем сетей прямого распространения с рассредоточенным временем запаздывания.

Раздел 13.10 завершит главу несколькими заключительными замечаниями.

13.2. Структуры кратковременной памяти

Основной задачей памяти является *преобразование статической сети в динамическую*. В частности, внедряя память в структуру статической сети (например, многослойного персептрона), мы делаем выходной сигнал зависимым от времени. Этот подход к созданию нелинейных динамических систем достаточно прост, так как обеспечивает четкое разделение обязанностей: статическая сеть учитывает нелинейность, а память — время.

Кратковременная память[3] может быть реализована в непрерывном и дискретном времени. Непрерывное время обозначим символом t, а дискретное — символом n. Цепь, показанная на рис. 13.1 и состоящая из сопротивления и емкости, представляет собой пример памяти непрерывного времени, которая характеризуется импульсным откликом $h(t)$, экспоненциально убывающим по времени t. Эта цепь отвечает за память на синаптическом уровне и может быть представлена аддитивной моделью нейрона, которая будет описана далее в этой главе. В этом же разделе мы сконцентрируем основное внимание на памяти дискретного времени.

Полезным инструментом при работе с системами дискретного времени является так называемое *z-преобразование*. Пусть $\{x(n)\}$ — некоторая последовательность в дискретном времени, которая может быть расширена в бесконечно далекое прошлое. z-преобразование такой последовательности определяется следующим образом:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}, \tag{13.1}$$

где z^{-1} — *оператор единичной задержки* (unit delay operator). Это значит, что оператор z^{-1}, примененный к $x(n)$, формирует версию сигнала с задержкой $x(n-1)$. Предположим, что $x(n)$ применяется к системе дискретного времени, имеющей импульсный отклик $h(n)$. Тогда выход всей системы определяется *суммой свертки* (convolution sum):

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k). \tag{13.2}$$

[3] Структуры кратковременной памяти и их роль во временной обработке описаны в [758].

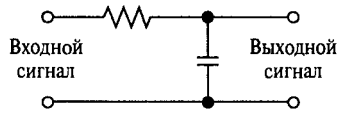


Рис. 13.1. Цепь, состоящая из емкости и сопротивления

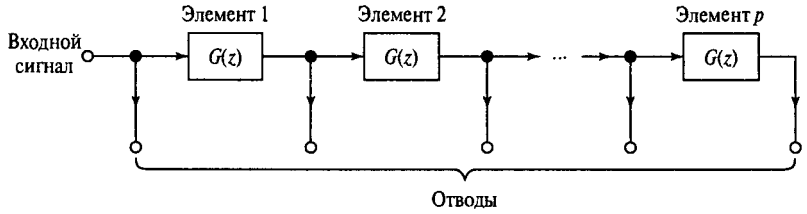


Рис. 13.2. Обобщенная память порядка p

Если $x(n)$ равно единичному импульсу, то $y(n)$ сводится к *импульсному отклику* (impulse response) $h(n)$ системы. Одним из важных свойств z-преобразований является то, что *свертка во временной области преобразовывается в произведение в z-области* [444], [802]. Если обозначить z-преобразование последовательностей $\{h(n)\}$ и $\{y(n)\}$ как $H(z)$ и $Y(z)$ соответственно, применение z-преобразования к формуле (13.2) приводит к соотношению

$$Y(z) = H(z)X(z), \tag{13.3}$$

или, эквивалентно:

$$H(z) = \frac{Y(z)}{X(z)}. \tag{13.4}$$

Функция $H(z)$ называется *передаточной функцией* (transfer function) системы.

На рис. 13.2 показана блочная диаграмма памяти дискретного времени, состоящей из p идентичных блоков с каскадным соединением. Далее значение p будем называть *порядком* (order) памяти.

Если рассматривать каждый из блоков задержки как оператор, его можно описать передаточной функцией $G(z)$. Также каждый блок можно описать в терминах импульсного отклика $g(n)$, который обладает следующими свойствами.

- Он является *причинным* (causal), т.е. $g(n) = 0$ для $n < 0$.
- Он является *нормализованным* (normalized), т.е. $\sum_{n=0}^{\infty} |g(n)| = 1$.

Исходя из этого, $g(n)$ называют *образующим ядром* (generating kernel) памяти дискретного времени.

В свете рис. 13.2 можно формально определить кратковременную память как линейную, инвариантную ко времени, систему с одним входом и несколькими выходами (SIMO), образующее ядро которой удовлетворяет вышеописанным требованиям. Точки коммутации, к которым подключены выходные терминалы памяти, обычно называют *отводами* (tap). Обратите внимание, что память порядка p содержит $p + 1$ отвод (один отвод принадлежит входу).

Атрибуты любой структуры памяти определяются в терминах глубины и разрешения. Обозначим как $g_p(n)$ общий импульсный отклик памяти, определяемый как p последовательных сверток $g(n)$, или, эквивалентно, обратное z-преобразование $G^p(z)$. *Глубина памяти* (memory depth) D определяется как первый момент по времени $g_p(n)$:

$$D = \sum_{n=0}^{\infty} n g_p(n). \tag{13.5}$$

Память с небольшой глубиной D хранит информацию только за относительно короткий период времени, в то время как память с большой глубиной хранит информацию из более отдаленного прошлого. *Разрешение памяти* (memory resolution), обозначаемое символом R, определяется как число отводов структуры памяти, соответствующее единице времени. Эта величина определяет уровень грубости хранения информации: чем выше разрешение, тем более точно представлена информация. При фиксированном количестве отводов произведение глубины и разрешения памяти является константой, равной порядку памяти p.

Различные варианты образующего ядра $g_p(n)$ приводят к различным значениям глубины D и разрешения R, что будет продемонстрировано двумя нижеследующими примерами структур памяти.

Память на основе линии задержки с отводами

На рис. 13.3 показана блочная диаграмма простейшей и наиболее широко используемой формы кратковременной памяти, которая называется памятью на основе линии задержки с отводами. Она состоит из p операторов единичной задержки, каждый из которых характеризуется функцией $G(z) = z^{-1}$. Это значит, что образующее ядро $g(n) = \delta(n - 1)$, где $\delta(n)$ — единичный импульс:

$$\delta(n) = \begin{cases} 1, & n = 0, \\ 0, & n \neq 0. \end{cases} \tag{13.6}$$

Общий импульсный отклик памяти, показанной на рис. 13.3, равен $g_p(n) = \delta(n - p)$. Подставляя это значение $g_p(n)$ в (13.5), получим глубину памяти, равную p, что интуитивно понятно. На рис. 13.3 показано только по одному отводу в единицу

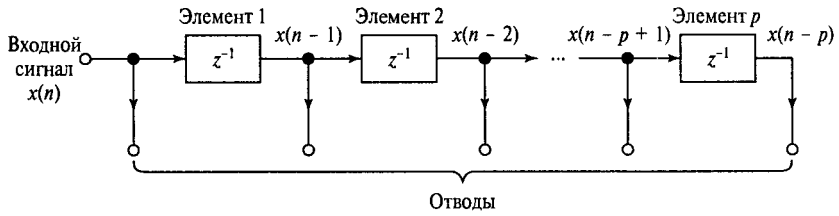


Рис. 13.3. Обычная память на основе линии задержки с отводами порядка p

времени. Исходя из этого, разрешение такой памяти $R = 1$. Таким образом, глубина памяти на основе линии задержки с отводами возрастает линейно пропорционально порядку p, при этом ее разрешение фиксировано и равно единице. Произведение глубины на разрешение дает в результате константу p.

Для того чтобы осуществлять управление глубиной памяти, требуется дополнительная степень свободы. Это обеспечивается моделью, альтернативной линии задержки с отводами, которую мы рассмотрим ниже.

Гамма-память

На рис. 13.4 показан граф передачи сигнала основного функционального блока $G(z)$, используемого в структуре, называемой *гамма-памятью* (Gamma memory) [256]. Каждый блок такой памяти состоит из цикла с единичной задержкой и подстраиваемым параметром μ. Передаточной функцией каждого такого блока является

$$G(z) = \frac{\mu z^{-1}}{1 - (1 - \mu)z^{-1}} = \frac{\mu}{z - (1 - \mu)}. \tag{13.7}$$

Для устойчивости единственный полюс $G(z)$ при $z = 1 - \mu$ должен лежать внутри единичного круга на z-плоскости. Это, в свою очередь, требует, чтобы

$$0 < \mu < 2. \tag{13.8}$$

Образующее ядро гамма-памяти является обратным z-преобразованием $G(z)$, т.е.

$$g(n) = \mu(1 - \mu)^{n-1}, n \geq 1. \tag{13.9}$$

Условие (13.8) гарантирует, что $g(n)$ экспоненциально стремится к нулю при стремлении n к бесконечности.

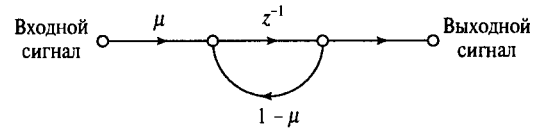


Рис. 13.4. Граф передачи сигнала в одном блоке гамма-памяти

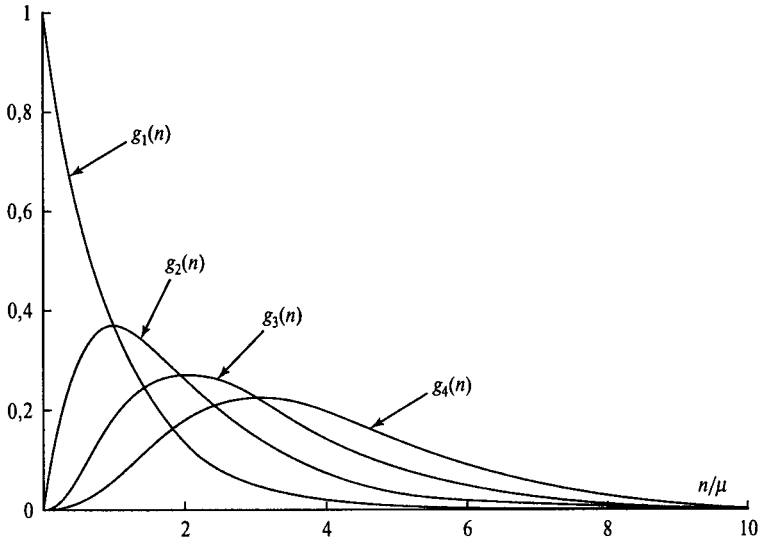


Рис. 13.5. Семейство импульсных откликов гамма-памяти для порядка $p = 1, 2, 3, 4$ и $\mu = 0,7$

Общий импульсный отклик гамма-памяти является обратным z-преобразованием общей передаточной функции:

$$G_p(z) = \left(\frac{\mu}{z - (1 - \mu)}\right)^p,$$

т.е.

$$g_p(n) = \binom{n-1}{p-1} \mu^p (1-\mu)^{n-p}, \quad n \geq p, \tag{13.10}$$

где (:) — биномиальный коэффициент, определяемый как $\binom{n}{p} = \frac{n(n-1)\ldots(n-p+1)}{p!}$ для целочисленных значений n и p. Общий импульсный отклик $g_p(n)$ для переменного p представляет собой дискретную версию подынтегрального выражения *гамма-функции* [256], откуда память и получила свое название. На рис. 13.5 показано семейство импульсных откликов $g_p(n)$, нормализованных по отношению к μ. Такое масштабирование привело к позиционированию пикового значения $g_p(n)$ в $n = p$.

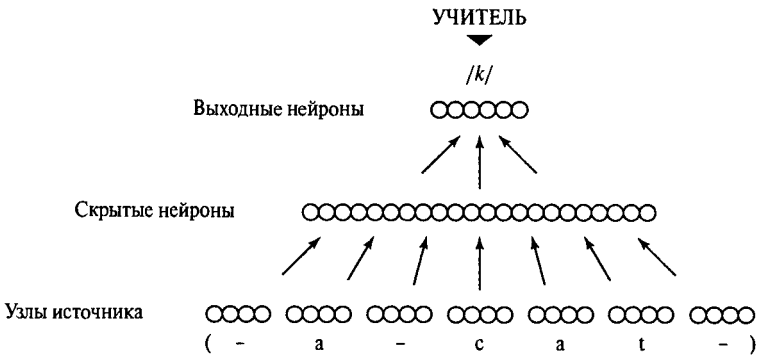


Рис. 13.6. Архитектура сети NETtalk

Глубина гамма-памяти равна p/μ, а ее разрешение — μ. При этом произведение глубины на разрешение равно p. Следовательно, выбирая μ меньшим единицы, гамма-память увеличивает свою глубину (огрубляя при этом разрешение) по сравнению с памятью на основе линии задержки с отводами при заданном порядке p. При $\mu = 1$ эти атрибуты сводятся к соответствующим значениям, определяемым памятью на основе линии задержки с отводам. Если μ превышает единицу, но меньше 2, то результат выражения $(1 - \mu)$ становится отрицательным, но имеющим абсолютное значение меньше единицы.

13.3. Сетевые архитектуры для временной обработки

Сетевая архитектура для временной обработки имеет несколько форм, собственно как и структуры памяти. В этом разделе речь пойдет о двух архитектурах сетей прямого распространения, которые описаны в литературе, посвященной временной обработке.

NETtalk

Архитектура NETtalk, предложенная в [962], впервые продемонстрировала огромную параллельную распределенную сеть, которая преобразовывала английскую разговорную речь в фонемы (*фонема* — это базовая лингвистическая единица). На рис. 13.6 показана схематическая диаграмма системы NETtalk, которая основана на многослойном персептроне с входным слоем, состоящим из 203 сенсорных узлов, 80 нейронов скрытого слоя и 26 нейронов выходного слоя. Все нейроны используют сигмоидальные (логистические) функции активации. Синаптические связи этой сети определяются в целом 18629 весами, включая *переменный порог* (threshold) для каждого нейрона, равный *смещению* (bias), взятому с обратный знаком. Для обучения сети использовался стандартный алгоритм обратного распространения.

Во входном слое этой сети содержалось семь групп узлов. Каждая группа кодировала одну букву входного текста. В любой момент времени на вход системы подавалось семь букв текста. Желаемый отклик для этого процесса обучения определялся как корректная фонема, ассоциируемая с центральной (т.е. четвертой) буквой семисимвольного окна. Оставшиеся шесть букв (по три с каждой стороны от центральной) являют собой частичный *контекст* для решения, принимаемого сетью. Окно перемещается по тексту на одну букву за шаг. На каждом шаге процесса сеть вычисляет фонему, а после каждого слова синаптические веса сети корректируются в соответствии с тем, насколько близко вычисленное произношение находится к корректному.

Сеть NETtalk обнаружила ряд сходств с наблюдаемой процедурой работы человека [962].

- Обучение подчиняется степенному закону.
- Чем больше количество слов, на которых обучается сеть, тем лучше обобщение и более корректно произношение новых слов.
- Производительность сети понижается достаточно медленно при повреждении отдельных синаптических весов сети.
- Повторное обучение после повреждения происходит значительно быстрее, нежели первичное.

Архитектура NETtalk блестяще продемонстрировала многие аспекты обучения, которое начинается со значительных "предварительных" знаний о входных примерах, которые постепенно повышаются на практике при преобразовании англоязычной речи в фонемы. Однако эта система не нашла практического применения.

Нейронные сети с задержкой по времени

Популярной сетью, которая для временной обработки использует обычные задержки, является так называемая *нейронная сеть с задержкой по времени* (time delay neural network — TDDN), которая впервые была описана в [613] и [1107]. TDDN — это многослойная сеть прямого распространения, скрытые и выходные нейроны которой *реплицируются по времени* (replicated across time). В этой работе было предложено с помощью спектрограммы извлекать в явном виде принципы симметрии, которая встречается в обособленном слове (фонеме). *Спектрограмма* (spectrogram) — это двумерный рисунок, в котором вертикальное измерение соответствует частоте, а горизонтальное — времени. Интенсивность (яркость) точек на спектрограмме соответствует энергии сигнала. На рис. 13.7, *а* показана версия TDDN с одним скрытым слоем [613]. Входной слой состоит из 192 (16×12) сенсорных узлов, кодирующих спектрограмму. Скрытый слой содержит 10 копий 8-ми скрытых нейронов, а выходной — 6 копий 4-х нейронов. Различные копии скрытого нейрона применяют одно и

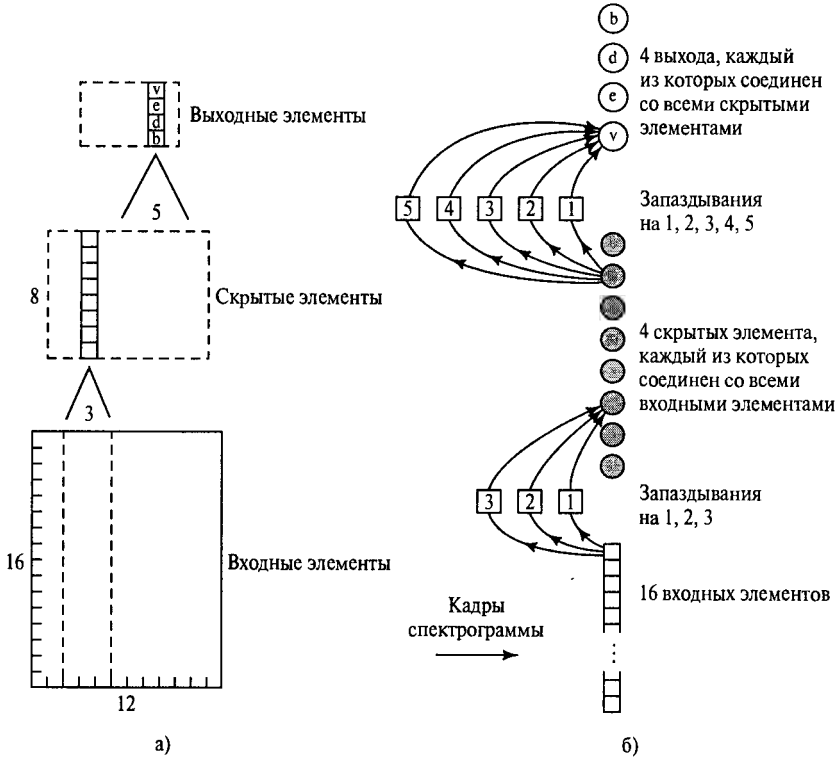


Рис. 13.7. Сеть, скрытые и выходные нейроны которой реплицированы во времени (а). Представление нейронной сети с задержкой во времени (TDNN) (б) (Приведено с разрешения авторов работы [613])

то же множество синаптических весов к узким (шириной в 3 шага по времени) окнам спектрограммы. Аналогично, различные копии выходного нейрона применяют узкие (шириной в 5 шагов по времени) окна псевдоспектрограммы, вычисленной скрытым слоем. На рис. 13.7, *б* представлена интерпретация *задержки по времени* (time delay) реплицируемой нейросети, изображенной на рис. 13.7, *а* (отсюда и берет свое название "нейронная сеть с задержкой по времени"). Эта сеть содержит в совокупности 544 синаптических веса. В [613] сеть TDDN использовалась для распознавания четырех обособленных слов "bee", "dee", "ee" и "vee", которые обрабатывались четырьмя выходными нейронами (см. рис. 13.7). При тестировании на данных, отличающихся от данных обучения, был получен результат распознавания, составляющий 93%. При более комплексном изучении [1107] для распознавания трех обособленных слов "bee", "dee" и "gee" использовалась сеть TDDN с двумя скрытыми слоями. При оценке производительности, в которой принимали участие три различных диктора, был получен результат 98,5%.

Оказалось, что сеть TDDN лучше работает при классификации временных образов, состоящих из последовательности векторов признаков с фиксированными размерностями (например, фонем). Однако при практическом распознавании речи нереально предполагать, что речевой сигнал будет разделен на обособленные фонемы, которые его составляют. Вместо этого приходится моделировать сегментированную временную структуру образцов речи. В частности, распознаватель речи должен работать с сегментами слов и предложений, длина и временная структура которых нелинейно варьируется. Для моделирования этих естественных характеристик традиционный подход к распознаванию речи может использовать структуру перехода состояний, подобную скрытой модели Маркова [514], [865]. В своей основе *скрытая модель Маркова* (hidden Markov model — HMM) представляет собой стохастический процесс, генерируемый рассматриваемой цепью Маркова и множеством распределений наблюдений, ассоциированным со скрытыми слоями. В литературе описано множество гибридов сетей TDDN и HMM[4].

13.4. Фокусированные сети прямого распространения с задержкой по времени

При *структурном распознавании образов* (structural pattern recognition) принято использовать статические нейронные сети. В противоположность этому *временное распознавание образов* (temporal pattern recognition) требует обработки образов, изменяющихся во времени, и генерации отклика в конкретный момент времени, который зависит не только от текущего, но и от нескольких предыдущих его значений. На рис. 13.8 показана блочная диаграмма *нелинейного фильтра*, созданного на основе статической нейронной сети [758]. Эта сеть стимулируется посредством кратковременной памяти. В частности, для заданного входного сигнала, состоящего из текущего $(x(n))$ и p предыдущих значений $(x(n-1),\ldots,x(n-p))$, хранимых в *памяти линейной задержки* (delay line memory) порядка p, свободные параметры сети корректируются с целью минимизации среднеквадратической ошибки между выходом этой сети $y(n)$ и желаемым откликом $d(n)$.

---

[4] Использование гибридов TDDN и HMM при распозиавании речи описано в [120], [143], [544].

Некоторые такие гибриды объединяют кодировщик кадров на базе TDDN (т.е. отображение "детектора акустических признаков" на "классы слов-предложений") и определитель маршрута слово-предложение (word/sentence path finder) HMM (т.е. отображение "символа фонемы" на "классы слов-предложений"), где обе составляющие действуют независимо друг от друга. В некоторых более сложных гибридах TDDN-HMM функция квадратичной ошибки потерь для всей системы используется таким образом, что потери, связанные с ошибкой слово–предложение, могут быть минимизированы. Примером последней схемы может служить TDDN с множеством состояний (multi-state TDDN), описанная в [405], [406]. Простейший гибрид раздельно спроектированных модулей часто приводит к несоответствию производительностей обучения и тестирования. В этом отношении сети TDDN с множеством состояний зарекомендовали себя лучше.

В фундаментальном смысле рекуррентные сети (которые будут рассматриваться в главе 15) имеют большую способность моделировать временную структуру речевого сигнала, чем репликационные сети типа TDDN. Тем не менее из-за существенной нестационарности и нелинейности речевого сигнала даже рекуррентных сетей может быть недостаточно для точного распознавания речи.

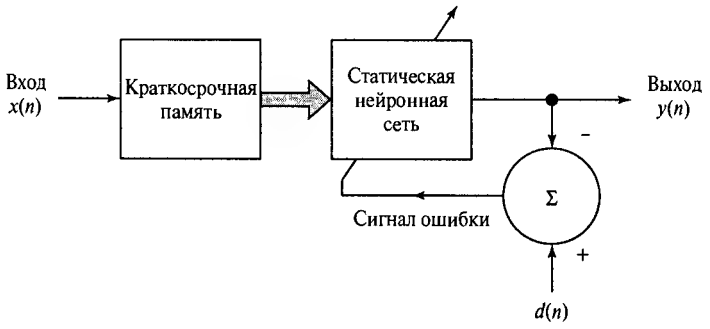


Рис. 13.8. Нелинейный фильтр, созданный на статической нейронной сети

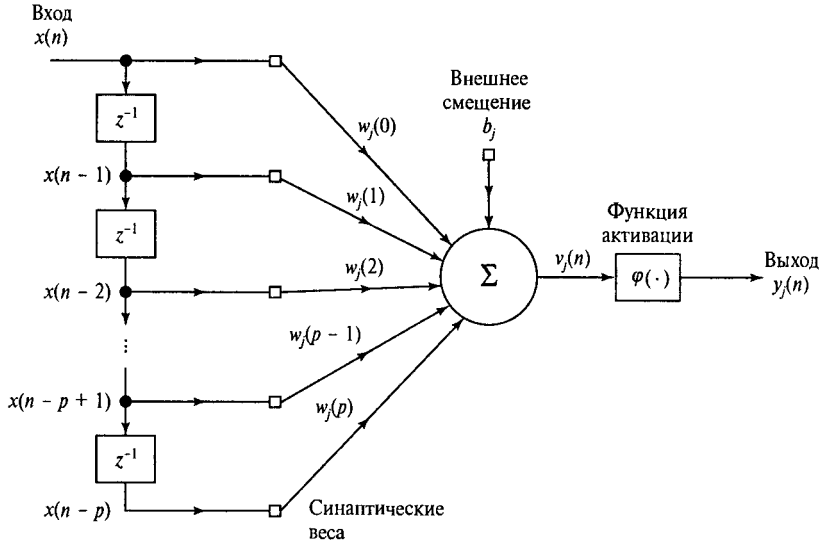


Рис. 13.9. Фокусированный нейронный фильтр

Структура, показанная на рис. 13.8, может быть реализована на уровне как обособленного нейрона, так и сети нейронов. Эта два варианта показаны на рис. 13.9 и 13.10.

Для упрощения выкладок на рис. 13.9, 13.10 в качестве структуры кратковременной памяти использовалась память на основе линии задержки с отводами. Естественно, оба этих рисунка можно обобщить, используя элемент памяти с передаточной функцией $G(z)$, а не z^{-1}.

Элемент временной обработки на рис. 13.9 состоит из памяти на основе линии задержки с отводами, отводы которой соединены с синапсами нейрона. Эта память извлекает временную информацию, содержащуюся во входном сигнале, которая, в свою очередь, передается синаптическими весами в нейрон. Элемент обработки на рис. 13.9 называют *фокусированным нейронным фильтром* (focused neuronal filter). Здесь под понятием фокусировки подразумевается концентрация всей структуры па-

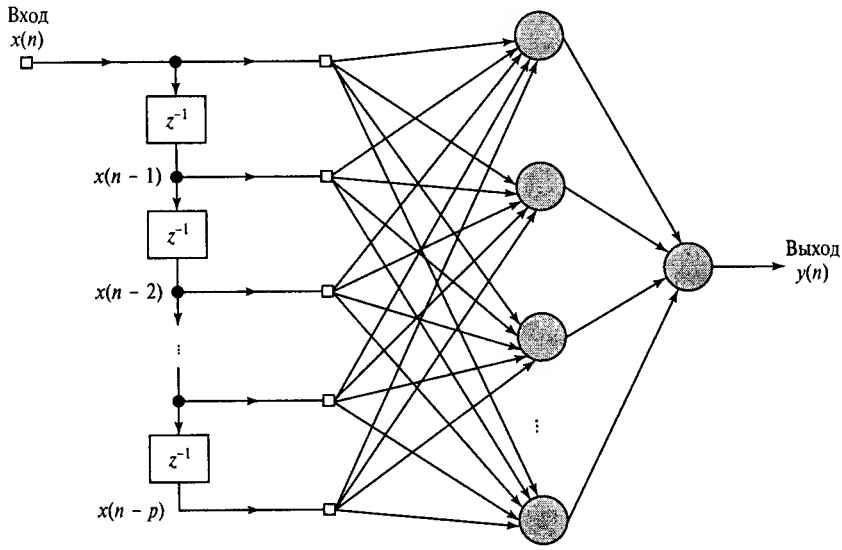


Рис. 13.10. Фокусированная сеть прямого распространения с задержкой по времени (TLFN). Для упрощения представления уровни смещения (bias) опущены

мяти на *переднем плане* (front end) элемента. Выход этого фильтра в ответ на входной сигнал $x(n)$ и его предыдущие значения $x(n-1),\ldots,x(n-p)$ вычисляется по следующей формуле:

$$y_j(n) = \varphi\left(\sum_{l=0}^{p} w_j(l)x(n-l) + b_j\right), \tag{13.11}$$

где $\varphi(\cdot)$ — функция активации нейрона j; $w_j(l)$ — его синаптические веса; b_j — его *смещение* (bias). Обратите внимание, что вход функции активации состоит из суммы смещения и свертки последовательностей входных сигналов и синаптических весов нейрона.

Переходя к рис. 13.10, на котором показана *фокусированная сеть прямого распространения с задержкой по времени* (focused time lagged feedforward network — TLFN), мы видим более мощный нелинейный фильтр, состоящий из памяти на основе линии задержки с отводами порядка p и многослойного персептрона. Для обучения этого фильтра можно использовать стандартный алгоритм обратного распространения (см. главу 4). В момент времени n "временной образ" применяется к входному слою сети в виде вектора сигнала:

$$x(n) = [x(n), x(n-1), \ldots, x(n-p)]^T,$$

который можно рассматривать как описание *состояния* нелинейного фильтра в момент времени n. Одна эпоха состоит из последовательности состояний, количество которых определяется порядком памяти p и мощностью множества примеров обучения N.

Выход нелинейного фильтра (предполагается, что многослойный персептрон имеет единственный скрытый слой, как показано на рис. 13.10) задается формулой

$$y(n) = \sum_{j=1}^{m_1} w_j y_j(n) = \sum_{j=1}^{m_1} w_j \varphi \left(\sum_{l=0}^{p} w_j(l) x(n-l) + b_j \right) + b_0. \qquad (13.12)$$

Здесь предполагается линейность выхода фокусированной TLFN, синаптические веса составляют множество $\{w_j\}_{j=1}^{m_1}$, где m_1 — размерность скрытого слоя, а смещение обозначено символом b_0.

13.5. Компьютерное моделирование

В этом компьютерном эксперименте будет исследована возможность использования фокусированной TLFN (см. рис. 13.10) для моделирования временного ряда, представляющего следующий сложный сигнал с частотной модуляцией:

$$x(n) = \sin(n + \sin(n^2)), n = 0, 1, 2, \ldots.$$

Сеть используется в качестве *одношаговой системы прогнозирования* (one-step predictor), в которой $x(n+1)$ содержит желаемый отклик на входной сигнал, состоящий из множества $\{x(n-l)\}_{l=0}^{p}$. Эта сеть имеет следующие характеристики.

Порядок памяти на основе линии задержки с отводами, p	20
Скрытый слой, m_1	10 нейронов
Функции активации скрытых нейронов	Логистические
Выходной слой	1 нейрон
Функция активации выходного нейрона	Линейная
Параметр интенсивности обучения (для обоих слоев)	0,01
Константа момента	Отсутствует

Множество данных, использованных для обучения сети, состояло из 500 случайных примеров, каждый их которых состоял из 20 упорядоченных по времени примеров, выбранных из последовательностей $\{x(n)\}$.

На рис. 13.11, *а* показана суперпозиция одношагового прогнозирования, выполняемого сетью на не встречавшихся ранее (при обучении) данных и фактического сигнала. На рис. 13.11, *б* показан график ошибки прогнозирования, определяемой как разность между прогнозированным и фактическим значениями. Среднеквадратическое значение этой ошибки прогнозирования равно $1,2 \times 10^{-3}$.

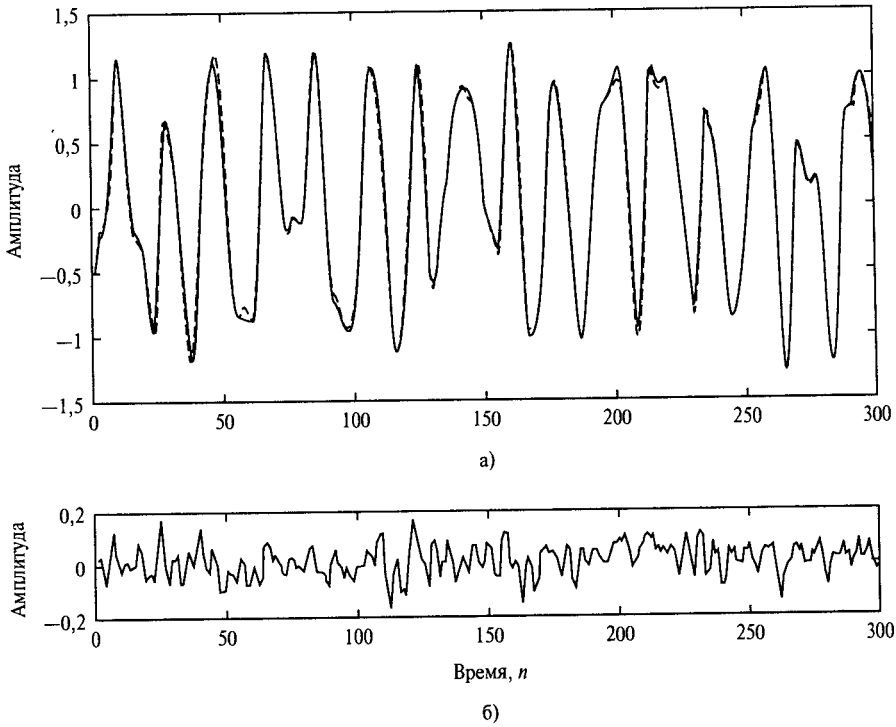


Рис. 13.11. Результаты компьютерного моделирования с одношаговым прогнозированием: суперпозиция фактического (сплошная линия) и прогнозированного (пунктир) сигналов (а), ошибка прогнозирования (б)

13.6. Универсальная теорема миопического отображения

Нелинейный фильтр (см. рис. 13.9) можно привести к виду, показанному на рис. 13.12. Эта обобщенная динамическая структура состоит из двух функциональных блоков. Блок с меткой $\{h_j\}_{j=1}^{L}$ представляет собой многочисленные свертки во временной области, т.е. банк параллельно работающих *линейных фильтров*; h_j выбирается из большого множества ядер с действительными значениями, каждое из которых представляет собой импульсный отклик некоторого линейного фильтра. Блок с меткой **N** представляет собой некоторую нелинейную статическую (не имеющую памяти) сеть прямого распространения (например, многослойный персептрон). Структура, показанная на рис. 13.12, представляет собой *универсальный динамический оператор* (universal dynamic mapper). В [928] показано, что любое инвариантное к смещению *миопическое отображение* (myopic map) может быть при достаточно мягких условиях как угодно хорошо равномерно аппроксимировано структурой, имеющей форму, показанную на рис. 13.12. Требование миопичности сети эквивалентно *равномерному затуханию па-*

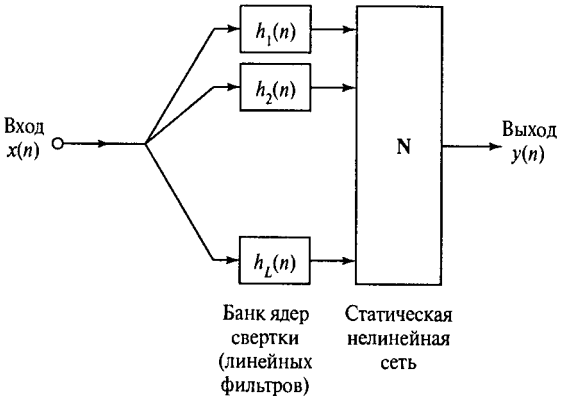


Рис. 13.12. Общая структура для универсальной теоремы миопического отображения

мяти (uniform fading memory). При этом предполагается, что преобразование имеет свойство *причинности* (causal), т.е. данный выходной сигнал генерируется отображением в момент времени $n \geq 0$ только в том случае, если соответствующий входной сигнал применяется в момент времени $n = 0$. Под инвариантностью к смещению подразумевается следующее: если $y(n)$ — выход отображения входного сигнала $x(n)$, тогда выход отображения смещенного входного сигнала $x(n-n_0)$ будет равен $y(n-n_0)$, где n_0 — некоторое *натуральное* (integer) число. В [929] показано, что для любого инвариантного к смещению отображения одной переменной, имеющего равномерно затухающую память, существует некоторая гамма-память и статическая нейронная сеть, комбинация которых произвольно хорошо аппроксимирует это отображение.

Теперь можно сформулировать *универсальную теорему миопического отображения*[5] [928], [929].

Любое инвариантное к смещению миопическое динамическое отображение может быть как угодно хорошо аппроксимировано структурой, состоящей из двух функциональных блоков: статической нейронной сети и банка линейных фильтров, формирующего ее входные данные.

Структура, внедренная в эту теорему, может иметь форму фокусированной TLFN. Также важно отметить, что эта теорема выполняется и в случаях, когда входной и выходной сигналы являются функциями конечного числа переменных (как, например, при обработке изображений).

Универсальная теорема миопического отображения имеет широкое практическое применение. Она не только осуществляет математическое обоснование NETtalk и его возможного расширения, использующего гамма-память, но и формулирует среду для создания более сложных моделей нелинейных динамических процессов. Многочис-

[5] Основные формулировки и доказательства универсальной теоремы миопического отображения содержатся в [927].

ленные свертки на переднем плане структуры (см. рис. 13.12) могут быть реализованы линейными фильтрами с импульсными откликами конечной (FIR) или бесконечной (IIR) длительности. С другой стороны, статическая нейронная сеть может быть реализована с использованием многослойного персептрона, сети на основе радиальных базисных функций или машины опорных векторов и соответствующих алгоритмов (см. главы 4–6). Другими словами, при создании нелинейных фильтров и моделей нелинейных динамических процессов можно опираться на материал, изложенный в главах, посвященных обучению с учителем. Более того, предполагая устойчивость линейных фильтров, входящих в структуру, приведенную на рис. 13.12, можно говорить о *внутренней устойчивости* (inherently stable) всей этой структуры. Таким образом, имеется четкое разделение ролей и методов работы с кратковременной памятью и нелинейностью, не содержащей память.

13.7. Пространственно-временные модели нейрона

Фокусированный нейронный фильтр (см. рис. 13.9) имеет довольно интересную интерпретацию, о которой речь пойдет в этом разделе. Комбинацию элементов единичной задержки и соответствующих синаптических весов можно рассматривать как *фильтр с конечной импульсной характеристикой* (finite-duration impulse response filter — FIR) порядка p (рис. 13.13, *а*). FIR-фильтр является одним из основных конструктивных блоков при обработке цифрового сигнала [444], [802]. Следовательно, фокусированный нейронный фильтр (см. рис. 13.9) фактически является нелинейным FIR-фильтром, показанным на рис. 13.13, *б*. Можно построить такое представление и таким образом расширить вычислительную мощность нейрона в пространственном смысле, используя все m_0 входы (рис. 13.14). Пространственно-временная модель, приведенная на рис. 13.14, называется *нейронным фильтром с несколькими входами* (multiple input neuronal filter).

Существует еще один способ описания модели, показанной на рис. 13.14: ее можно рассматривать как *распределенный нейронный фильтр* (distributed neuronal filter) в том смысле, что действия фильтрации проводятся для различных пространственных точек. Эта пространственно-временная модель имеет следующие характеристики.

- Нейрон имеет m_0 "первичных" синапсов, каждый из которых состоит из линейного фильтра дискретного времени, выполненного в форме порядка p. Первичные синапсы при обработке сигнала учитывают пространственное измерение.
- Каждый из "первичных" синапсов имеет $(p + 1)$ "вторичных" синапсов, которые связывают его с соответствующими входами и отводами памяти FIR-фильтра, учитывая при обработке сигнала временное измерение.

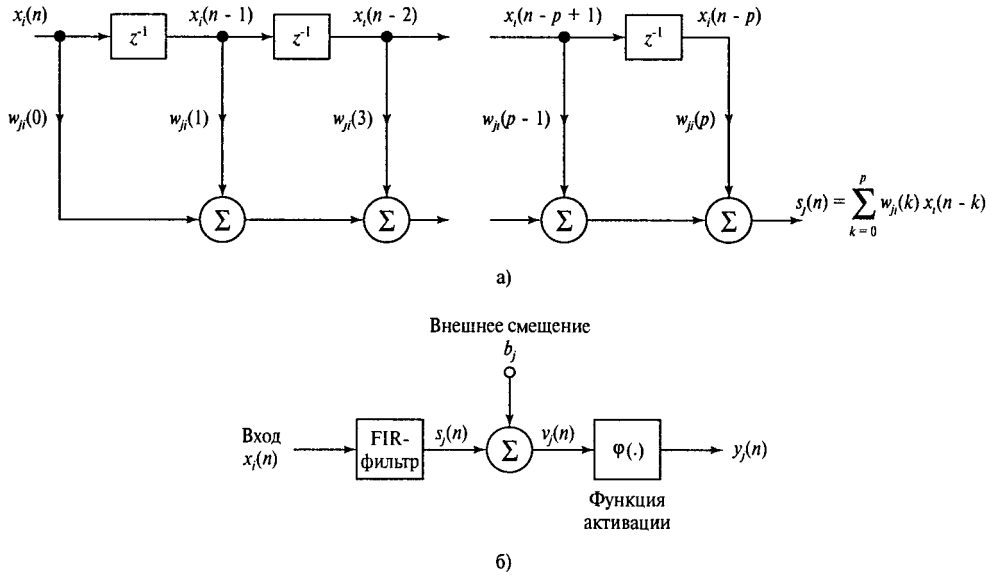


Рис. 13.13. Фильтр с конечной импульсной характеристикой (FIR) (а); интерпретация нейронного фильтра как нелинейного FIR-фильтра (б)

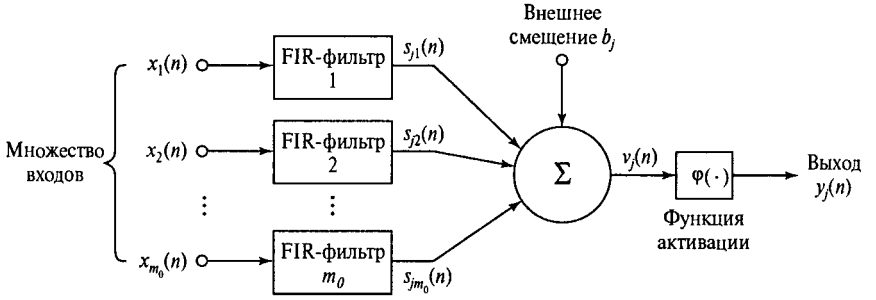


Рис. 13.14. Нейронный фильтр с несколькими входами

Синаптическая структура нейронного фильтра (см. рис. 13.14) напоминает дерево, показанное на рис. 13.15. Общее количество синаптических весов в этой структуре равно $m_0(p+1)$.

Переходя к математическим терминам, пространственно-временную обработку, выполняемую нейронным фильтром на рис. 13.14, можно описать следующим образом:

$$y_j(n) = \varphi\left(\sum_{i=1}^{m_0}\sum_{l=0}^{p} w_{ji}(l)x_i(n-l) + b_j\right), \qquad (13.13)$$

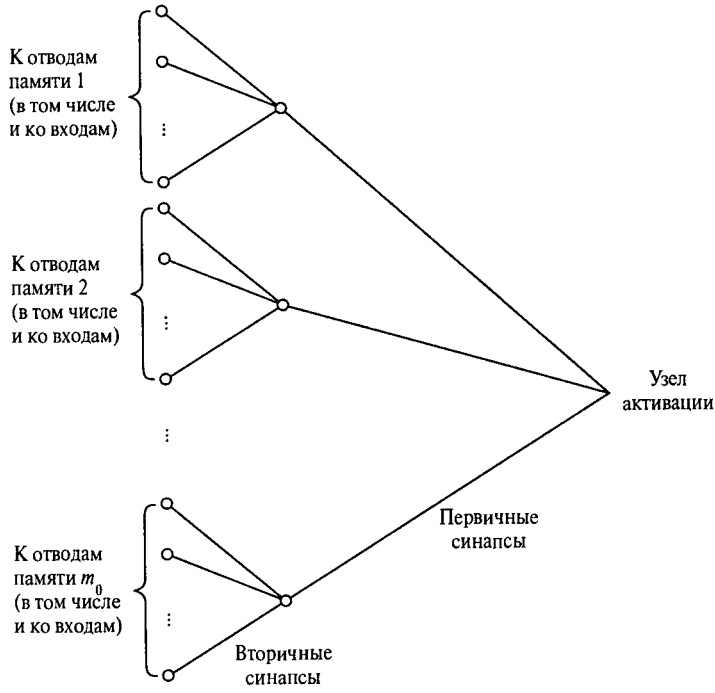


Рис. 13.15. Древоподобное описание синаптической структуры нейронного фильтра с несколькими входами

где $w_{ji}(l)$ — вес l-го вторичного синапса, принадлежащего i-му первичному синапсу; $x_i(n)$ — входной сигнал, применяемый к i-му первичному синапсу в момент времени n; b_j — смещение (bias), применяемое к нейрону. Индуцированное локальное поле $v_j(n)$ нейрона, т.е. аргумент функции активации φ в выражении (13.13), можно рассматривать как дискретную по времени аппроксимацию следующей формулы в непрерывном времени:

$$v_j(t) = \sum_{i=l}^{m_0} \int_{-\infty}^{t} h_{ji}(\lambda) x_i(t-\lambda) d\lambda + b_j. \tag{13.14}$$

Интеграл в формуле (13.14) является *сверткой* непрерывного входного сигнала $x_i(t)$ и импульсного отклика $h_{ij}(t)$, характеризующей линейный непрерывный фильтр, представляющий синапс i. Уравнение (13.14) является самым общим способом описания пространственно-временного поведения индуцированного локального поля нейрона.

Аддитивная модель

Уравнение (13.14) является основой другой широко используемой пространственно-временной модели нейрона. Предположим, например, что мы упростили временное поведение нейрона, использовав параметр масштабирования для определения знака и направления "типичного" синаптического импульсного отклика. В этом случае можно записать, что

$$h_{ji}(t) = w_{ji} \cdot h_j(t) \text{ для всех } i, \tag{13.15}$$

где $h_j(t)$ моделирует временные характеристики типичного постсинаптического потенциала, а w_{ji} — скаляр, определяющий его знак (возбуждение или торможение) и общую мощность соединения между нейроном j и входом i [968]. Таким образом, подставив (13.15) в (13.14), а затем изменив порядок интегрирования и суммирования, получим:

$$\begin{aligned} v_j(t) &= \int_{-\infty}^{t} h_j(\lambda) \left(\sum_{i=1}^{m_0} w_{ji} x_i(t-\lambda) \right) d\lambda + b_j = \\ &= h_j(t) * \left(\sum_{i=1}^{m_0} w_{ji} x_i(t) \right) + b_j, \end{aligned} \tag{13.16}$$

где символ звездочки обозначает операцию свертки. Форма общего импульсного отклика $h_j(t)$ зависит от объема требуемой детализации. Чаще всего для него используется следующая функция:

$$h_j(t) = \frac{1}{\tau_j} \exp\left(-\frac{t}{\tau_j}\right), \tag{13.17}$$

где τ_j — некоторая *временная константа*, являющаяся характеристикой нейрона j. Функция времени $h_j(t)$ в выражении (13.17) сходна с импульсным откликом простой электрической цепи, состоящей из сопротивления R_j и емкости C_j, соединенных параллельно и питающихся от текущего источника, т.е.

$$\tau_j = R_j C_j. \tag{13.18}$$

Следовательно, выражения (13.16) и (13.17) можно использовать при формулировке модели, показанной на рис. 13.16. Говоря физическими терминами, синаптические веса $w_{j1}, w_{j2}, \ldots, w_{jm_0}$ представлены проводимостью (т.е. величиной, обратной сопротивлению), а соответствующие входы $x_1(t), x_2(t), \ldots, x_{m_0}(t)$ представлены потенциалами (т.е. напряжением). Сумматор характеризуется малым входным сопро-

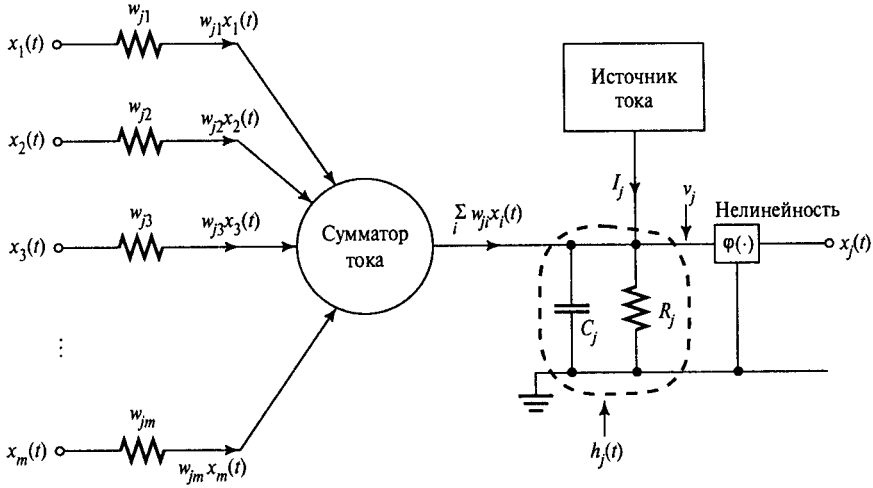


Рис. 13.16. Аддитивная модель нейрона

тивлением, единичным усилением тока и высоким выходным сопротивлением. Это значит, что он выступает как узел суммирования входящих токов. Таким образом, общий ток, проходящий через эту *резисторно-емкостную цепь* (resistance-capacitance — RC), составляет:

$$\sum_{i=1}^{m_0} w_{ji}x_i(t) + I_j,$$

где первое слагаемое (суммирование) соответствует возбуждениям $x_1(t), x_2(t), \ldots, x_{m_0}(t)$, воздействующим на синаптические веса (проводимости) $w_{j1}, w_{j2}, \ldots, w_{jm_0}$, а второе слагаемое является током источника I_j, представляющий внешнее смещение (bias) b_j.

В литературе по нейронным сетям модель, показанную на рис. 13.16, обычно называют *аддитивной* (additive model). Эту модель можно рассматривать как *неоднородную* (lumped) аппроксимацию электрической цепью модели в виде *распределенной линии передачи* (distributed transmission line model) *биологического дендрического нейрона* (biological dendritic neuron) [867]. Такую природу RC-цепи (см. рис. 13.16) можно также объяснить тем фактом, что сам биологический синапс является фильтром, предназначенным для хорошей аппроксимации [954].

13.8. Распределенные сети прямого распространения с задержкой по времени

Универсальный алгоритм миопического отображения, который обеспечивает математическое обоснование фокусированной TLFN, ограничен только теми отображениями, которые инвариантны к смещению. Значение этого ограничения состоит в том, что фокусированные TLFN применимы только в стационарных (инвариантных по времени) средах. Это ограничение можно обойти, используя *распределенные сети прямого распространения с задержкой по времени* (TLFN). Здесь под распределенностью подразумевается то, что неявное влияние времени распределено по всей сети. Конструкция такой сети основывается на применении нейронного фильтра с несколькими входами (см. рис. 13.14) в качестве пространственно-временной модели нейрона.

Обозначим символом $w_{ji}(l)$ вес, соединенный с l-м отводом FIR-фильтра, моделирующего синапс, соединяющий выход нейрона i с нейроном j. Индекс l варьируется от нуля до порядка FIR-фильтра — p. Согласно этой модели, сигнал $s_{ji}(n)$, образующийся на выходе i-го синапса нейрона j, представляется *суммой свертки* (convolution sum):

$$s_{ji}(n) = \sum_{l=0}^{p} w_{ji}(l)x_i(n-l), \tag{13.19}$$

где n — дискретное время. Выражение (13.19) можно переписать в матричном виде, если ввести следующие определения *вектора состояния* и *вектора весов* для синапса i:

$$\mathbf{x}_i(n) = [x_i(n), x_i(n-1), \ldots, x_i(n-p)]^T, \tag{13.20}$$

$$\mathbf{w}_{ji}(n) = [w_{ji}(0), w_{ji}(1), \ldots, w_{ji}(p)]^T. \tag{13.21}$$

Таким образом, скалярный сигнал $s_{ji}(n)$ можно выразить как скалярное произведение векторов $\mathbf{w}_{ji}(n)$ и $\mathbf{x}_i(n)$:

$$s_{ji}(n) = \mathbf{w}_{ji}^T\mathbf{x}_i(n). \tag{13.22}$$

Формула (13.22) определяет выход $s_{ji}(n)$ i-го синапса нейрона j в модели, показанной на рис. 13.14, в ответ на подачу входного вектора $\mathbf{x}_i(n)$, где $i = 1, 2, \ldots, m_0$. Вектор $\mathbf{x}_i(n)$ называют "состоянием", так как он представляет состояние i-го синапса в момент времени n. Исходя из этого, суммируя вклад всего множества, состоящего из m_0 синапсов модели (т.е. суммируя по всем индексам i), можно описать выход $y_j(n)$ нейрона j следующей парой уравнений:

$$v_j(n) = \sum_{i=1}^{m_0} s_{ji}(n) + b_j = \sum_{i=1}^{m_0} \mathbf{w}_{ji}^T\mathbf{x}_i(n) + b_j, \tag{13.23}$$

$$y_j(n) = \varphi(v_j(n)), \tag{13.24}$$

где $v_j(n)$ — индуцированное локальное поле нейрона j; b_j — внешнее смещение (bias); $\varphi(\cdot)$ — нелинейная функция активации нейрона. Предполагается, что во всех нейронах сети используется одна и та же форма нелинейности. Обратите внимание, что если векторы состояний и весов ($\mathbf{w}_{ji}$ и $\mathbf{x}_i(n)$) заменить соответствующими скалярами (w_{ji} и x_i) и если операцию скалярного произведения заменить простым умножением, то динамическая модель нейрона, представленная выражениями (13.23) и (13.24), сведется к статической модели, описанной в главе 4.

13.9. Алгоритм обратного распространения во времени

Для обучения распределенной сети TLFN требуется некоторый алгоритм обучения с учителем, в котором фактический отклик всех нейронов выходного слоя сравнивается с желаемым (целевым) откликом в каждый момент времени. Предположим, что нейрон j находится в выходном слое сети, а его выход обозначен как $y_j(n)$, при этом желаемый отклик этого нейрона обозначен $d_j(n)$ (имеются в виду значения в момент времени n). После этого можно определить *мгновенное* (instantaneous) *значение* суммы среднеквадратических ошибок сети следующим образом:

$$\mathbf{E}(n) = \frac{1}{2}\sum_j e_j^2(n), \tag{13.25}$$

где индекс j соответствует только нейронам выходного слоя, а $e_j(n)$ — сигнал ошибки, определяемый следующим образом:

$$e_j(n) = d_j(n) - y_j(n). \tag{13.26}$$

Нашей целью является минимизация *функции стоимости* (cost function), определяемой как сумма квадратичных ошибок $\mathbf{E}(n)$ по всем моментам времени:

$$E_{\text{общая}} = \sum_n \mathbf{E}(n). \tag{13.27}$$

Алгоритм, который можно использовать для вычисления оценки оптимального вектора весов и с помощью которого достигается цель, основан на аппроксимации метода наискорейшего спуска.

Очевидно, что для решения поставленной задачи нужно продифференцировать функцию стоимости (13.27) по вектору весов $\mathbf{w}_{ji}$:

$$\frac{\partial \mathbf{E}_{\text{общая}}}{\partial \mathbf{w}_{ji}} = \sum_n \frac{\partial \mathbf{E}(n)}{\partial \mathbf{w}_{ji}}. \tag{13.28}$$

Продолжая далее действия в соответствии с подходом мгновенного градиента, мы *раскрываем сеть во времени* (unfold the network in time). Стратегия состоит в следующем: в первую очередь нужно попытаться устранить в сети все задержки по времени, разворачивая ее в эквивалентную "статическую", но более громоздкую сеть, после чего можно применить стандартный алгоритм обратного распространения для вычисления мгновенных градиентов ошибки. К сожалению, такой подход имеет следующие недостатки.

- Потери в смысле симметрии между прямым распространением состояний и обратным распространением в терминах, необходимых для вычисления мгновенных градиентов ошибки.
- Отсутствие удобных рекурсивных формул для распространения ошибки.
- Потребность в глобальной бухгалтерии для ведения учета того, какие статические веса остались неизменными в эквивалентной сети, полученной *раскрытием* (unfolding) распределенной TLFN.

Несмотря на то что использование мгновенной оценки градиента является очевидным подходом при разработке временной версии обратного распространения, с практической точки зрения он не рационален.

Чтобы обойти проблемы, связанные с подходом мгновенного градиента, в [1110], [1111] было предложено действовать следующим образом. Во-первых, мы понимаем, что раскрытие общей ошибки градиента в сумму мгновенных ошибок (см. (13.28)) не является единственно возможным. В частности, можно рассмотреть следующий альтернативный способ выражения частной производной функции стоимости $\mathbf{E}_{\text{общая}}$ по вектору весов $\mathbf{w}_{ji}$:

$$\frac{\partial \mathbf{E}_{\text{общая}}}{\partial \mathbf{w}_{ji}} = \sum_{n} \frac{\partial \mathbf{E}_{\text{общая}}}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial \mathbf{w}_{ji}}, \tag{13.29}$$

где индекс времени n касается только $v_j(n)$. Можно интерпретировать частные производные $\partial \mathbf{E}_{\text{общая}}/\partial v_j(n)$ как изменение функции стоимости, вызванное изменением индуцированного локального поля $v_j(n)$ нейрона j в момент времени n. Здесь важно обратить внимание на то, что

$$\frac{\partial \mathbf{E}_{\text{общая}}}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial \mathbf{w}_{ji}} \neq \frac{\partial \mathbf{E}(n)}{\partial \mathbf{w}_{ji}}.$$

Равенство достигается только тогда, когда берется сумма по всем n (см. (13.28) и (13.29)).

Имея разложение (13.29), можно использовать идею градиентного спуска в пространстве весов. В частности, можно реализовать следующую рекурсию для коррек-

ции вектора весов-отводов $\mathbf{w}_{ji}(n)$:

$$\mathbf{w}_{ji}(n+1) = \mathbf{w}_{ji}(n) - \eta \frac{\partial \mathbf{E}_{\text{общая}}}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial \mathbf{w}_{ji}(n)}, \tag{13.30}$$

где η — *параметр скорости обучения* (learning-rate parameter). Из определения (13.23) видно, что для любого нейрона j сети частная производная индуцированного локального поля $v_j(n)$ по отношению к $\mathbf{w}_{ji}(n)$ определяется следующим образом:

$$\frac{\partial v_j(n)}{\partial \mathbf{w}_{ji}(n)} = \mathbf{x}_i(n), \tag{13.31}$$

где $\mathbf{x}_i(n)$ — входной вектор, применяемый к синапсу i нейрона j. Более того, можно определить *локальный градиент* (local gradient) нейрона j следующим образом:

$$\delta_j(n) = -\frac{\partial \mathbf{E}_{\text{общая}}}{\partial v_j(n)}. \tag{13.32}$$

Следовательно, равенство (13.30) можно переписать в знакомом виде:

$$\mathbf{w}_{ji}(n+1) = \mathbf{w}_{ji}(n) + \eta \delta_j(n) \mathbf{x}_i(n). \tag{13.33}$$

Как и при выводе стандартного алгоритма обратного распространения (см. главу 4), явная форма локального градиента $\delta_j(n)$ зависит от того, где находится нейрон j: в выходном или скрытом слое сети. Оба этих случая мы рассмотрим ниже.

Случай 1. Нейрон j является выходным элементом

Для выходного слоя имеем:

$$\delta_j(n) = \frac{\partial \mathbf{E}_{\text{общая}}}{\partial v_j(n)} = -\frac{\partial \mathbf{E}(n)}{\partial v_j(n)} = e_j(n) \varphi'(v_j(n)), \tag{13.34}$$

где $e_j(n)$ — сигнал ошибки, измеряемый на выходе нейрона j; $\varphi'(\cdot)$ — производная функции активации $\varphi(\cdot)$ по своему аргументу.

Случай 2. Нейрон j является скрытым элементом

Если нейрон j находится в скрытом слое, символом $\mathbf{A}$ можно обозначить множество всех нейронов, входы которых получают сигнал от выхода нейрона j при прямом распространении. Пусть $v_r(n)$ — индуцированное локальное поле нейрона r, принадлежащего множеству $\mathbf{A}$. Тогда можно записать, что

$$\delta_j(n) = -\frac{\partial \mathbf{E}_{\text{общая}}}{\partial v_j(n)} = -\sum_{r \in \mathbf{A}} \sum_k \frac{\partial \mathbf{E}_{\text{общая}}}{\partial v_r(k)} \frac{\partial v_r(k)}{\partial v_j(n)}. \tag{13.35}$$

Подставляя определение (13.32) (в котором индекс r используется вместо индекса j) в (13.35), можно записать:

$$\delta_j(n) = \sum_{r \in \mathbf{A}} \sum_k \delta_r(k) \frac{\partial v_r(k)}{\partial v_j(n)} = \sum_{r \in \mathbf{A}} \sum_k \delta_r(k) \frac{\partial v_r(k)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)}, \tag{13.36}$$

где $y_j(n)$ — выход нейрона j. Далее несложно заметить, что частная производная $\partial y_j(n)/\partial v_j(n)$ равна $\varphi'(v_j(n))$, если нейрон j находится вне множества **A**. Таким образом, этот множитель можно вынести за рамки двойного суммирования и переписать (13.36) следующим образом:

$$\delta_j(n) = \varphi'(v_j(n)) \sum_{r \in \mathbf{A}} \sum_k \delta_r(k) \frac{\partial v_r(k)}{\partial y_j(n)}. \tag{13.37}$$

Как было определено ранее, $v_r(n)$ обозначает индуцированное локальное поле нейрона r, питаемого выходом нейрона j. Адаптируя выражения (13.19) и (13.23) к данной ситуации, $v_r(k)$ можно выразить следующим образом:

$$v_r(k) = \sum_{j=0}^{m_0} \sum_{l=0}^{p} w_{rj}(l) y_j(n-l). \tag{13.38}$$

В равенство (13.38) включено смещение (bias) b_r, применяемое к нейрону r как слагаемое, соответствующее j =0:

$$w_{r0}(l) = b_r y_0(n-l) = 1 \text{ для всех } l \text{ и } n. \tag{13.39}$$

Индекс p, определяющий верхний предел внутреннего суммирования в (13.38), является порядком каждого из синаптических фильтров нейрона r и всех остальных нейронов рассматриваемого слоя. Индекс m_0, определяющий верхний предел внешнего суммирования в (13.38), является общим числом первичных синапсов, относящихся к нейрону r. Признавая, что сумма свертки по l является коммутативной, соотношение (13.38) можно переписать в следующем виде:

$$v_r(k) = \sum_{j=0}^{m_0} \sum_{l=0}^{p} y_j(l) w_{rj}(n-l). \tag{13.40}$$

Дифференцируя (13.40) по y_j, получим:

$$\frac{\partial v_r(k)}{\partial y_j(n)} = \begin{cases} w_{rj}(k-l), & n \le k \le n+p, \\ 0 \text{ в } & \text{— противном случае.} \end{cases} \tag{13.41}$$

В свете (13.41) частные производные $\partial v_r(k)/\partial y_j(n)$ в (13.37), для которых k находится вне диапазона $n \leq k \leq n+p$, равны нулю. Если нейрон j является скрытым, подставив (13.41) в (13.37), получим:

$$\begin{aligned}\delta_j(n) &= \varphi'(v_j(n)) \sum_{r \in \mathbf{A}} \sum_{k=n}^{n+p} \delta_r(k) w_{rj}(k-l) \\ &= \varphi'(v_j(n)) \sum_{r \in \mathbf{A}} \sum_{l=0}^{p} \delta_r(n+l) w_{rj}(n). \end{aligned} \tag{13.42}$$

Определим новый вектор размерности $(p+1) \times 1$:

$$\boldsymbol{\Delta}_r(n) = [\delta_r(n), \delta_r(n+1), \ldots, \delta_r(n+p)]^T. \tag{13.43}$$

Ранее с помощью выражения (13.21) уже был определен вектор весов $\mathbf{w}_{ji}$. Используя матричные символы, выражение (13.42) можно переписать в более компактном виде:

$$\delta_j(n) = \varphi'(v_j(n)) \sum_{r \in \mathbf{A}} \boldsymbol{\Delta}_r^T(n) \mathbf{w}_{rj}, \tag{13.44}$$

где $\boldsymbol{\Delta}_r^T(n)\mathbf{w}_{rj}$ — скалярное произведение векторов $\boldsymbol{\Delta}_r(n)$ и $\mathbf{w}_{rj}$, которые имеют размерность $(p+1)$. Равенство (13.44) является окончательным выражением оценки $\delta_j(n)$ для нейрона j скрытого слоя.

Теперь можно записать окончательное выражение для коррекции весов в методе *обратного распространения во времени* (temporal back propagation) [1110], [1111]:

$$\mathbf{w}_{ji}(n+1) = \mathbf{w}_{ji}(n) + \eta \delta_j(n) \mathbf{x}_i(n). \tag{13.45}$$

$$\delta_j(n) = \begin{cases} e_j(n)\varphi'(v_j(n)), & \text{нейрон } j \text{ принадлежит выходному слою,} \\ \varphi'(v_j(n)) \sum\limits_{r \in \mathbf{A}} \boldsymbol{\Delta}_r^T(n) \mathbf{w}_{rj}, & \text{нейрон } j \text{ принадлежит скрытому слою.} \end{cases} \tag{13.46}$$

Эти выражения можно обобщить для любого количества скрытых слоев. Мы ясно видим, что эти соотношения представляют собой *векторное обобщение* стандартного алгоритма обратного распространения. Если заменить входной вектор весов $\mathbf{x}_i(n)$, вектор весов $\mathbf{w}_{rj}$ и вектор локального градиента $\boldsymbol{\Delta}_r$ соответствующими скалярами, алгоритм обратного распространения во времени сведется к обычному алгоритму обратного распространения (см. главу 4).

Для вычисления $\delta_j(n)$ для нейрона j, расположенного в скрытом слое, нужно *распространить* (propagate) δs из следующего слоя назад через синаптические фильтры, возбуждение которых выводится из нейрона j в соответствии с формулой (13.44). Этот механизм обратного распространения показан на рис. 13.17. Таким образом, локальный градиент $\delta_j(n)$ формируется не просто взвешенным суммированием, а обратной

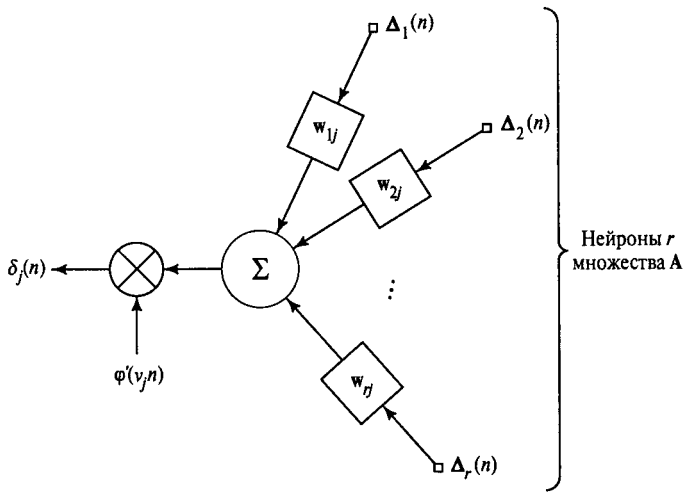


Рис. 13.17. Обратное распространение локальных градиентов в распределенной сети TLFN

фильтрацией каждым первичным синапсом. В частности, каждое новое множество векторов входных сигналов и желаемых откликов порождает новый шаг прямой и обратной фильтрации.

Теперь видны практические преимущества использования временного алгоритма обратного распространения. Среди них можно назвать следующие.

1. Сохраняется симметрия между прямым распространением состояний и обратным распространением ошибок. Таким образом, поддерживается принцип параллельной распределенной обработки.
2. Каждый отдельный вес синаптического фильтра используется только один раз – при вычислении δs. Таким образом, в методе мгновенного градиента не наблюдается избыточных вычислений.

При выводе временного алгоритма обратного распространения[6], описанного формулами (13.45) и (13.46), предполагается, что веса синаптических фильтров фиксированы для всех вычислений градиента. Естественно, это не совсем корректное предположение для реальной адаптации. Следовательно, возникают несоответствия в производительности между временным алгоритмом обратного распространения и временной версией, полученной с использованием метода *мгновенного* (instantaneous) градиента. Тем не менее эти несоответствия обычно имеют незначительный характер. При небольшом значении параметра интенсивности обучения η расхождение в характеристиках обучения этих двух алгоритмов при их практическом применении будет незначительным.

[6] Альтернативный схематический вывод алгоритма обратного распространения во времени содержится в [1112].

Ограничения причинности

При внимательном изучении формулы (13.42) оказывается, что вычисление $\delta_j(n)$ не носит *причинный* характер, так как требует знания будущих значений δs и $\mathbf{w}$. Для того чтобы сделать эти вычисления причинными, в первую очередь обратим внимание на то, что точная ссылка на время, используемая для адаптации, не важна. Более того, синаптическая структура сети состоит из FIR-фильтров. Следовательно, причинность требует использования дополнительной буферизации для хранения внутренних состояний сети. Отсюда следует, что нужно требовать, чтобы адаптация всех векторов весов основывалась только на текущем и прошлых значениях сигналов ошибки. Таким образом, можно сразу же определить $\delta_j(n)$ для нейрона j в выходном слое и адаптировать веса синаптических фильтров этого слоя. Для слоя, предшествующего выходному, ограничения причинности предполагают, что вычисление локального градиента для нейрона j этого слоя

$$\delta_j(n-p) = \varphi'(v_j(n-p)) \sum_{r \in \mathbf{A}} \boldsymbol{\Delta}_r^T(n-p)\mathbf{w}_{rj}, \tag{13.47}$$

основывается только на текущем и предыдущих значениях вектора Δ_j:

$$\boldsymbol{\Delta}_r(n-p) = [\delta_r(n-p), \delta_r(n+1-p), \ldots, \delta_r(n)]^T. \tag{13.48}$$

Уравнение (13.47) получено из второй строки (13.46) заменой индекса n на $(n-p)$, где p — порядок каждого из синаптических FIR-фильтров. Как уже говорилось ранее, состояния $\mathbf{x}_i(n-p)$ должны храниться, чтобы можно было вычислить произведение $\delta_i(n-p)\mathbf{x}_i(n-p)$ для адаптации вектора весов, соединяющих нейрон j последнего скрытого слоя с нейроном i предыдущего слоя. Для сети с несколькими крытыми слоями можно продолжить эту операцию еще для одного скрытого слоя (т.е. отстоящего на два слоя от выходного в обратном направлении), удвоив смещение по времени. Операция продолжается подобным образом для всех вычислительных слоев сети.

Исходя из этого, можно сформулировать *причинную* (causal) форму временного алгоритма обратного распространения, которая описана в табл. 13.1.

Несмотря на то что представленный алгоритм менее эстетичен, чем его непричинная форма, описываемая выражениями (13.45), (13.46), эти формы алгоритма отличаются друг от друга только заменой индексов.

Подводя итог, можно утверждать следующее.

- Сигнал δs распространяется по слоям сети (в обратном направлении), слой за слоем, *без* дополнительных задержек. Такой стиль распространения заставляет внутренние значения δs смещаться по времени.
- Для корректного смещения по времени эти состояния (т.е. значения $\mathbf{x}_i(n)$) сохраняются в форме соответствующих произведений, необходимых для адаптации век-

ТАБЛИЦА 13.1. Алгоритм обратного распространения во времени

1. Входной сигнал распространяется по сети в прямом направлении, слой за слоем. Определяется сигнал ошибки $e_j(n)$ для нейрона j выходного слоя путем вычитания фактического выходного сигнала из желаемого отклика. Также записываются векторы состояний для всех синапсов сети.
2. Для нейрона j выходного слоя вычислим:
$$\delta_j(n) = e_j(n)\varphi'_j(n); \quad \mathbf{w}_{ji}(n+1) = \mathbf{w}_{ji}(n) + \eta\delta_j(n)\mathbf{x}_i(n),$$
где $\mathbf{x}_j(n)$ — состояние i-го синапса скрытого нейрона, соединенного с выходным нейроном j.
3. Для нейрона j скрытого слоя вычислим:
$$\delta_j(n-lp) = \varphi'(v_j(n-lp))\sum_{r\in\mathbf{A}}\boldsymbol{\Delta}_r^T(n-lp)\mathbf{w}_{rj},$$
$$\mathbf{w}_{ji}(n+1) = \mathbf{w}_{ji}(n) + \eta\delta_j(n-lp)\mathbf{x}_i(n-lp),$$
где p — порядок каждого синаптического FIR-фильтра, а индекс l идентифицирует рассматриваемый скрытый слой. В сетях с несколькими скрытыми слоями $l = 1$ соответствует скрытому слою, непосредственно предшествующему выходному слою, $l = 2$ соответствует скрытому слою, отстоящему от выходного на два слоя, и т.д.

торов весов. Другими словами, добавленные задержки хранения требуются только для состояний, в то время как обратное распространение дельт осуществляется без задержек.

- Обратное распространение δs остается симметричным по отношению к прямому распространению состояний.
- Порядок вычислений линеен по отношению к нумерации синаптических весов в сети, как и при подходе мгновенного градиента.

Распределенные сети TLFN являются более сложной структурой, чем фокусированные TLFN, описанные в разделе 13.4. Более того, временной алгоритм обратного распространения, требуемый для обучения распределенных TLFN, требует бо́льших вычислительных мощностей, чем стандартный алгоритм обратного распространения, используемый для обучения фокусированных TLFN. При окончательном анализе использование одного из этих двух подходов определяется тем, требует ли рассматриваемая задача временной обработки в стационарной или нестационарной среде[7].

[7] В [1110] временной алгоритм обратного распространения использовался для нелинейного прогнозирования нестационарных временных последовательностей, представляющих хаотические пульсации в NH_3-лазере. Эти последовательности были частью соревнования Santa Fe Institute Time-series Competition, которое проводилось в США в 1992 году. Решение Вана (Wan) для этой задачи временной обработки выиграло это соревнование у разнообразных решений, использовавших стандартные сети прямого распространения и рекуррентные сети, а также некоторые традиционные линейные техники [1110]. Хаос будет рассмотрен в главе 14.

13.10. Резюме и обсуждение

Потребность во временной обработке возникает в многочисленных прикладных задачах, в том числе следующих.

- *Прогнозирование* (prediction) и *моделирование* временных рядов (time series) [145], [434].
- *Шумоподавление* (noise cancellation), в котором требуется использовать первичные сенсоры (получающие желаемый сигнал, загрязненный шумом) и *ссылочные сенсоры* (reference sensor) (получающие коррелированную версию шума) [434], [1144].
- *Адаптивное уравнивание* (adaptive equalization) неизвестного канала связи [434], [859].
- *Адаптивное управление* (adaptive control) [773].
- *Идентификация систем* (system identification) [664].

Уже существуют отработанные теории для решения вышеперечисленных задач для случаев, когда рассматриваемая система или физический механизм линейны (см. вышеуказанные книги). Однако, если система или физический механизм нелинейны, задача оказывается более сложной. В ситуациях такого рода нейронные сети имеют потенциал для предоставления подходящего решения, таким образом проявляя свои отличительные характеристики.

В контексте нейронных сетей для временной обработки подходят два их типа.

- *Сети прямого распространения с задержкой по времени* (TLFN).
- *Рекуррентные сети.*

Обсуждение рекуррентных сетей приводится в следующих двух главах. В настоящей главе описаны два типа сетей TLFN — *фокусированные* и *распределенные.* В фокусированных TLFN кратковременная память сосредоточена на переднем плане статической сети, что упрощает ее конструкцию. Обучение фокусированной TLFN осуществляется стандартным алгоритмом обратного распространения (в предположении, что статическая сеть реализована многослойным персептроном). *Универсальная теорема миопического отображения* (universal myopic mapping theorem) [928], [929] является теоремой существования в том смысле, что она содержит математическое доказательство возможности аппроксимации произвольного миопического отображения (т.е. причинного отображения с равномерно затухающей памятью) с помощью последовательности двух функциональных блоков: банка линейных фильтров и статической нейросети. Такая структура может быть создана с помощью фокусированной TLFN. Эти сети и являются физической реализацией вышеназванной теоремы.

Второй класс TLFN, распределенные, основан на использовании пространственно-временной модели нейрона — нейронного фильтра с несколькими входами. Эта модель использует фильтры импульсного отклика с конечной длительностью (FIR-фильтры) в качестве синаптических фильтров. Как таковые нейронные фильтры с несколькими входами являются мощным функциональным блоком для пространственно-временной обработки сигнала, созданной на базе одного нейрона. Для обучения этих фильтров можно использовать LMS-алгоритм (см. главу 3), однако для обучения распределенной TLFN требуется более сложный алгоритм обучения. В качестве примера такого алгоритма в разделе 13.9 был сформулирован временной алгоритм обратного распространения. Отличительным свойством распределенных TLFN является способ, которым неявное представление времени распределяется по всей сети. Из этого и происходит их способность работать в нестационарной (изменяющейся во времени) среде. В противоположность этому в фокусированных TLFN по определению явное представление времени сконцентрировано на переднем плане сети, что ограничивает их использование только стационарными (инвариантными к времени) средами.

Задачи

Фокусированные TLFN

13.1. Перечислите основные атрибуты фокусированных TLFN, используемых для моделирования нелинейных динамических процессов.

13.2. Фокусированная TLFN, показанная на рис. 13.10, использует память на основе линии задержки с отводами для реализации кратковременной памяти. Каковы достоинства и недостатки фокусированных TLFN, использующих для кратковременной памяти гамма-память?

13.3. В главе 2 качественно описан динамический подход к реализации нелинейных адаптивных фильтров. Этот метод использует статические нейронные сети, моделирование которых обеспечивается путем подачи входных данных через скользящее окно. Это окно смещается при подаче каждого следующего примера данных. При этом самый старый пример отбрасывался, чтобы освободить место новому. Обсудите, как можно использовать фокусированные TLFN для реализации этой формы непрерывного обучения.

Пространственно-временны*е* модели нейронов

13.4. Рассмотрим нейронный фильтр, индуцированное локальное поле $v_j(t)$ которого определяется формулой (13.16). Предположим, что в этом равенстве функция времени $h(t)$ заменена смещенным единичным импульсом:

$$h_j(t) = \delta(t - \tau_j),$$

где τ_j — фиксированная задержка. Опишите, как изменится нейронный фильтр при такой модификации.

13.5. Используя алгоритм LMS, сформулируйте алгоритм обучения нейронного фильтра с несколькими входами, показанного на рис. 13.9.

Обратное распространение во времени

13.6. На рис. 13.18 показано использование *временного окна гауссовой формы* (Gaussian-shaped time window) как метода временно*й* обработки [139]. Временно*е* окно, ассоциированное с синапсом i нейрона j, обозначается как $\theta(n, \tau_{ji}, \sigma_{ji})$, где τ_{ji} и σ_{ji} — соответственно меры времени задержки и ширины:

$$\theta(n, \tau_{ji}, \sigma_{ji}) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{1}{\sigma_{ji}^2}(n - \tau_{ji})^2\right).$$

Исходя из этого, выход нейрона j моделируется следующим соотношением:

$$y_j(n) = \varphi\left(\sum_{i=0}^{m_0} w_{ji} u_i(n)\right),$$

где $u_i(n)$ — свертка входа $x_i(n)$ и временно*го* окна $\theta(n, \tau_{ji}, \sigma_{ji})$. Вес w_{ji} и задержка времени τ_{ji} синапса i, принадлежащего нейрону j, должны обучаться с учителем.

Это обучение может быть реализовано с использованием стандартного алгоритма обратного распространения. Продемонстрируйте этот процесс обучения с помощью вывода уравнений коррекции величин τ_{ji}, σ_{ji} и w_{ji}.

13.7. Материал, представленный в разделе 13.9, посвященном временно*му* алгоритму обратного распространения, имеет дело с синаптическими FIR-фильтрами равной длины. Как поступить, если FIR-фильтры будут иметь разную длину?

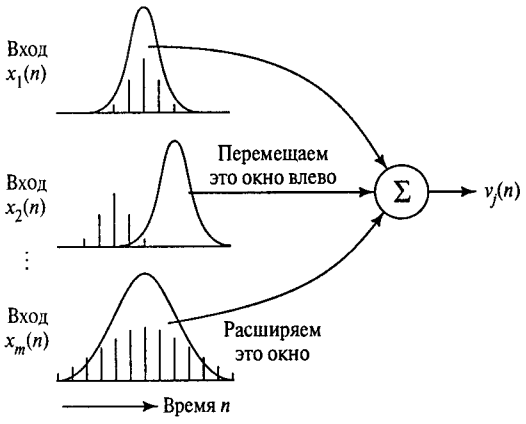


Рис. 13.18.

13.8. Обсудите, как временной алгоритм обратного распространения можно использовать для обучения распределенной TLFN для одношагового прогнозирования.

13.9. Расхождения между *причинной* (causal) и *беспричинной* (noncausal) формами алгоритма обратного распространения во времени аналогичны отличиям стандартного алгоритма LMS от алгоритма LMS с задержками во времени (см. главу 3). Продолжите эту аналогию.

Компьютерное моделирование

13.10. В этой задаче стандартный алгоритм обратного распространения будет использоваться для решения сложной задачи нелинейного прогнозирования и сравнивать его производительность с алгоритмом LMS. Рассматриваемые временные последовательности созданы с помощью дискретной *модели Вольтерра* (Volterra model), имеющей следующий вид:

$$x(n) = \sum_i g_i v(n-i) + \sum_i \sum_j g_{ij} v(n-i)v(n-j) + \ldots,$$

где $g_i, g_{ij}, \ldots$ — коэффициенты Вольтерра; $v(n)$ — образы белого, независимо распределенного гауссова шума; $x(n)$ — результирующий выход модели Вольтерра. Первое слагаемое суммирования представляет собой хорошо знакомую модель *ряда со скользящим средним* (moving average time series), а второе слагаемое двойной суммы — нелинейные компоненты все возрастающего порядка. В общем случае оценка коэффициентов Вольтерра довольно сложна, в основном по причине нелинейности соотношений по отношению к данным.

В данной задаче мы рассмотрим простейший пример:

$$x(n) = v(n) + \beta v(n-1)v(n-2).$$

Эта последовательность имеет нулевое среднее, она не коррелированна и, таким образом, имеет белый спектр. Однако примеры этой последовательности взаимозависимы. Таким образом, можно построить систему прогнозирования более высокого порядка. Дисперсия выхода этой модели определяется по следующей формуле:

$$\sigma_x^2 = \sigma_v^2 + \beta^2\sigma_v^2, \tag{13.49}$$

где σ_v^2 — дисперсия белого шума.

а) Сконструируйте многослойный персептрон, в котором входной слой состоит из 6 нейронов, скрытый — из 16 нейронов, а выходной содержит всего один нейрон. Для питания входного слоя используется память на основе линии задержки с отводами. Скрытые нейроны используют сигмоидные функции активации, ограниченные в интервале [0, 1], в то время как выходной нейрон работает в режиме линейного сумматора. Эта сеть обучается с помощью стандартного алгоритма обратного распространения, имеющего следующие характеристики.

Параметр интенсивности обучения	$\eta = 0,001$
Константа момента	$\alpha = 0,6$
Общее количество подаваемых примеров	100000
Количество примеров в одной эпохе	1000
Количество эпох	2500

Дисперсия белого шума σ_v^2 задана равной единице. Следовательно, при β =0,5 окажется, что дисперсия на выходе системы прогнозирования составит $\sigma_x^2 = 1,25$.

Вычислите кривую обучения этой системы прогнозирования, представив дисперсию выхода $x(n)$ как функцию от количества эпох примеров обучения, которое по условиям задачи ограничено числом 2500. Для подготовки каждой из эпох обучения используйте следующие два режима.

- Упорядочивание по времени примеров обучения при смене эпох производится по той же форме, по которой они генерируются.
- Упорядочивание примеров обучения по отношению к состояниям носит случайный характер.

Для мониторинга обучения системы прогнозирования используйте перекрестную проверку (см. главу 4), в которой множество проверки состоит из 1000 примеров.

б) Повторите этот эксперимент, используя алгоритм LMS, сконструированный для осуществления линейного прогнозирования на основе представленных шести примеров. Параметр скорости обучения в этом алгоритме установите в значение $\eta = 10^{-5}$.

в) Повторите весь эксперимент для следующих пар параметров: $\beta = 1$, $\sigma_v^2 = 2$; $\beta = 2$, $\sigma_v^2 = 5$.

Результаты каждого эксперимента должны показать, что в самом начале алгоритмы обратного распространения и LMS проходят практически одинаковый путь, после чего первый улучшает свою производительность прогнозирования, достигая заранее заданного значения дисперсии σ_x^2.

ГЛАВА 14

Нейродинамика

14.1. Введение

В предыдущей главе, посвященной обработке сигналов во времени, рассматривались структуры кратковременной памяти и работа статических нейронных сетей (на примере многослойного персептрона), выступающих в роли динамического оператора при поддержке структуры памяти. Еще одним важным способом встраивания времени в неявном виде в работу нейронной сети является использование *обратной связи* (feedback). Обратная связь может присутствовать в нейронной сети в двух видах: в виде *локальной обратной связи* (т.е. на уровне одного нейрона сети) и *глобальной обратной связи* (т.е. охватывающей всю сеть). Локальная обратная связь является объектом, с которым довольно просто работать, а глобальная обратная связь оказывает на сеть очень большое влияние. В литературе по нейронным сетям сети с одной или несколькими обратными связями называют *рекуррентными* (recurrent). В этой и следующей главах мы сосредоточим внимание на рекуррентных сетях, использующих глобальную обратную связь.

Обратная связь подобна обоюдоострому лезвию: будучи некорректно примененной, она может произвести пагубный эффект. В частности, применение обратной связи может привести к неустойчивости системы. В этой главе основной интерес представляет устойчивость рекуррентных сетей. Остальные аспекты этих сетей рассматриваются в следующей главе.

Область знаний, в которой нейронные сети рассматриваются как нелинейные динамические системы и основной упор делается на проблему *устойчивости* (stability), называется *нейродинамикой* (neurodinamics) [466]. Важным свойством устойчивости (или неустойчивости) нелинейных динамических систем является то, что оно распространяется на всю систему. В результате *наличие устойчивости всегда принимает одну из форм согласованности между отдельными частями системы* [72]. Изучение нейродинамики началось в 1938 году работой, в которой несбыточные мечты о применении динамики в биологии переросли в первое исследование [873].

Устойчивость нелинейных динамических систем является достаточно сложным вопросом. Когда речь идет о задаче устойчивости, обычно ее понимают в терминах

критерия устойчивости типа "ограниченный вход–ограниченный выход" (bounded input-bounded output stability criterion — BIBO). Согласно этому критерию, под устойчивостью понимается то, что выход системы *не должен неограниченно возрастать* в результате подачи ограниченного входного сигнала, начального состояния или нежелательных сбоев [159]. Критерий устойчивости BIBO хорошо подходит для линейных динамических систем. Однако бесполезно применять его к нейронным сетям, так как подобные нелинейные динамические системы являются BIBO-устойчивыми, поскольку в конструкцию нейрона встроена нелинейность с насыщением.

Когда речь идет об устойчивости в контексте нелинейных динамических систем, то подразумевается *устойчивость по Ляпунову* (stability in the sense of Lyapunov). В работе, написанной в 1892 году, Ляпунов (русский математик и инженер) представил фундаментальные концепции теории устойчивости, известные как *прямой метод Ляпунова* (direct method of Lyapunov). Этот метод широко используется для анализа устойчивости линейных и нелинейных систем, как зависящих, так и не зависящих от времени. Его можно напрямую применить к анализу устойчивости нейронных сетей. В связи с этим большая часть материала настоящей главы посвящена прямому методу Ляпунова. Следует заметить, что его применение — задача не из легких.

При изучении нейродинамики можно пойти двумя путями, в зависимости от интересующего ее применения.

- Путем *детерминированной нейродинамики*, в которой модель нейронной сети имеет детерминированное поведение. В математических терминах эта модель описывается как система *нелинейных дифференциальных уравнений*, определяющих точную эволюцию системы как функцию времени [202], [402], [479].
- Путем *статистической нейродинамики*, в которой модель нейронной сети подвержена влиянию помех. В этом случае работа ведется со *стохастическими нелинейными дифференциальными уравнениями*, и решение выражается в вероятностных терминах [25], [32], [825]. Комбинация стохастичности и нелинейности усложняет объект исследования.

В этой главе ограничимся детерминированной нейродинамикой.

Структура главы

Материал этой главы разбит на три части. В первой части (разделы 14.2–14.6) представлен вводный материал. В разделе 14.2 поданы некоторые фундаментальные концепции динамических систем, а в разделе 14.3 обсуждаются устойчивости точек равновесия. В разделе 14.4 описаны различные типы аттракторов, которые используются при изучении динамических систем. В разделе 14.5 мы вернемся к аддитивной модели нейрона, которая была выведена в главе 13. В разделе 14.6 описаны операции с аттракторами как нейросетевая парадигма.

Во второй части главы (разделы 14.7–14.11) речь пойдет об ассоциативной памяти. Раздел 14.7 посвящен подробной дискуссии о моделях Хопфилда и использовании их дискретных версий в качестве ассоциативной или контентно-адресуемой (content-addressable) памяти. В разделе 14.8 описано компьютерное моделирование этого применения сетей Хопфилда. В разделе 14.9 будет представлена теорема Кохена–Гроссберга для нелинейных динамических систем, для которой сети Хопфилда и другие виды ассоциативной памяти являются частным случаем. В разделе 14.10 описана еще одна нейродинамическая модель, которая хорошо подходит для кластеризации. В разделе 14.11 рассматривается компьютерное моделирование этой второй модели.

Последняя часть главы (разделы 14.12–14.14) посвящена вопросу хаоса. В разделе 14.12 рассмотрены инвариантные характеристики хаотического процесса, за чем в разделе 14.13 последует обсуждение вопроса, более тесно связанного с нашим предметом, — динамического восстановления хаотического процесса. В разделе 14.14 рассматривается компьютерное моделирование такого восстановления.

Глава завершится выводами и рассуждениями.

14.2. Динамические системы

Для того чтобы приступить к изучению нейродинамики, нужна *математическая модель*, описывающая динамику нелинейной системы. Для этой цели, естественно, подходит *модель в пространстве состояний* (state-space model). В соответствии с ней рассуждения будут проводиться в терминах *переменных состояний* (state variable), значения которых (в любой конкретный момент времени) должны содержать информацию, достаточную для прогнозирования эволюции системы. Пусть $x_1(t), x_2(t), \ldots, x_N(t)$ — переменные состояния некоторой нелинейной динамической системы, где t — *независимая переменная* непрерывного времени; N — *порядок* системы. Для удобства выкладок эти состояния собраны в вектор $\mathbf{x}(t)$ размерности $n \times 1$, который называется *вектором состояний* (state vector) системы. Динамика большого класса нелинейных динамических систем может быть представлена в форме системы дифференциальных уравнений первого порядка, имеющих вид

$$\frac{d}{dt}x_j(t) = F_j(x_j(t)), \quad j = 1, 2, \ldots, N, \tag{14.1}$$

где $F_j(\cdot)$ — некоторая нелинейная функция своего аргумента. Применяя векторные обозначения, всю систему дифференциальных уравнений можно представить в следующем, более компактном виде:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{F}(\mathbf{x}(t)), \tag{14.2}$$

где $\mathbf{F}$ — нелинейная вектор-функция, каждый элемент которой связан с соответствующим элементом вектора состояний:

$$\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_N(t)]^T. \tag{14.3}$$

Нелинейная динамическая система, в которой вектор-функция $\mathbf{F}(\mathbf{x}(t))$ не зависит явным образом от времени t, называется *автономной* (autonomic); в противном случае система называется *неавтономной*[1]. Мы рассмотрим только автономные системы.

Независимо от точной формы функции $\mathbf{F}(\mathbf{x}(t))$, вектор состояния может изменяться во времени t. В противном случае состояние $\mathbf{x}(t)$ будет константой, и система перестанет быть динамической. Таким образом, можно дать определение динамической системе.

Динамической является та система, состояния которой изменяются во времени.

Более того, $d\mathbf{x}/\mathrm{dt}$ можно представить себе как вектор скорости не в физическом, а в абстрактном смысле. Тогда, согласно (14.2), вектор-функцию $\mathbf{F}(\mathbf{x})$ можно назвать векторным полем скорости (velocity vector field), или просто полем скорости.

Пространство состояний

Целесообразнее рассматривать соотношение (14.2) как описывающее *движение* точки в N-мерном пространстве состояний. Это пространство состояний может быть Евклидовым, или подмножеством последнего. Это пространство может быть также и не-Евклидовым, например сферой, кругом, тором или любым другим дифференцируемым многообразием (differentiable manifold). Для дальнейшего обсуждения интерес представляет только Евклидово пространство.

Пространство состояний приобрело свою важность из-за того, что оно предоставляет визуально-концептуальное средство для анализа динамики нелинейной системы, описываемой уравнением (14.2). Оно фокусирует внимание на глобальных характеристиках движения, а не на отдельных аспектах аналитических или числовых решений этого уравнения.

В конкретный момент времени t наблюдаемое состояние системы (т.е. вектор состояний $\mathbf{x}(t)$) представлен одной точкой в N-мерном пространстве состояний. Изменение состояния системы с изменением времени представляется кривой в пространстве состояний, каждая точка которой (явно или неявно) является меткой времени наблюдения. Такая кривая называется траекторией (trajectory) или *орбитой* (orbit) системы. На рис. 14.1 показана траектория некоторой двумерной системы. Мгновенная скорость в этой траектории (т.е. вектор скорости $d\mathbf{x}(t)/\mathrm{dt}$) представлена вектором

[1] Неавтономная динамическая система определяется уравнением состояния: $\frac{d}{dt}\mathbf{x}(t) = \mathbf{F}(\mathbf{x}(t), t)$ с начальным состоянием $\mathbf{x}(t_0) = \mathbf{x}_0$. В неавтономных системах вектор поля $\mathbf{F}(\mathbf{x}(t), t)$ зависит от времени t. Таким образом, в отличие от автономных систем в общем случае нельзя принять начальное время равным нулю [818].

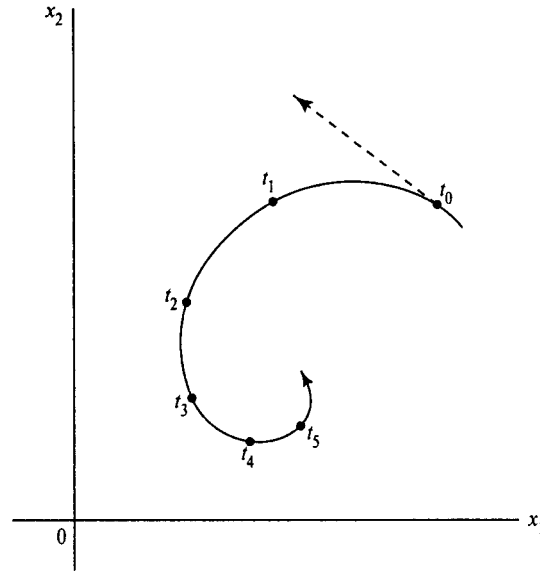


Рис. 14.1. Двумерная траектория (орбита) некоторой динамической системы

тангенса (tangent vector) (на рис. 14.1 он показан пунктирной линией для момента времени $t = t_0$). Аналогично можно найти вектор скорости для любой точки траектории.

Семейство траекторий, соответствующих различным начальным состояниям, называют портретом состояний (state portrait) системы. Этот портрет содержит точки в пространстве состояний, в которых определено векторное поле $\mathbf{F}(\mathbf{x})$. Обратите внимание, что в автономной системе через одно начальное состояние проходит только одна траектория. Полезным понятием, вытекающим из портрета состояний, является *поток* (flow) динамической системы, определяемый как движение пространства состояний в самом себе. Другими словами, можно представить себе, что пространство состояний течет в каждой своей точке, подобно жидкости, по определенной траектории [3]. Описанная здесь идея потока хорошо продемонстрирована на портрете состояний на рис. 14.2.

Для заданного портрета состояний динамической системы можно построить поле векторов скорости (тангенсов) — по одному вектору для каждой точки пространства. Таким образом, полученный рисунок, в свою очередь, является изображением векторного поля системы. На рис. 14.3 показано множество векторов скорости, формирующих картину общего вида этого поля. Полезность векторного поля заключается в том факте, что оно дает визуальное описание внутренних тенденций динамической системы к перемещению в любой точке пространства состояний.

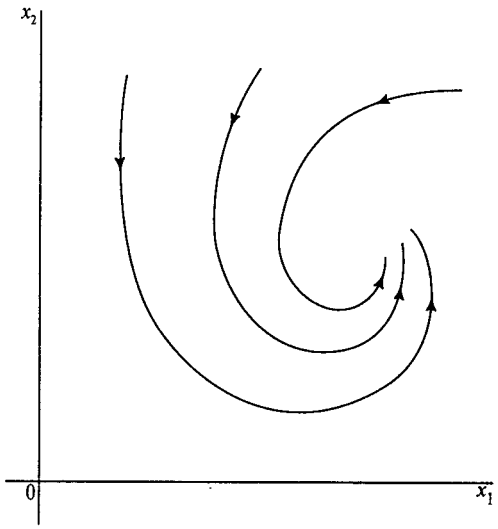


Рис. 14.2. Двумерный портрет состояний (фазовый портрет) некоторой динамической системы

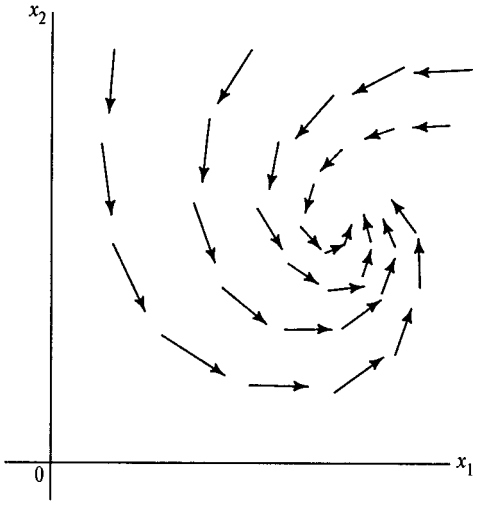


Рис. 14.3. Двумерное векторное поле некоторой динамической системы

Условие Липшица

Для того чтобы уравнение (14.2) в пространстве состояний имело решение, а также для единственности такого решения, на вектор-функцию $\mathbf{F}(\mathbf{x})$ необходимо наложить некоторые ограничения. Для упрощения выкладок нужно отказаться от зависимости вектора $\mathbf{x}$ от времени t. К этой практике мы будем иногда прибегать и в дальнейшем. Для того чтобы решение существовало, достаточно, чтобы вектор-функция $\mathbf{F}(\mathbf{x})$ была непрерывна по всем своим аргументам. Однако это условие не гарантирует единственности решения. Для этого необходимо наложить дополнительное ограничение,

известное как условие Липшица. Пусть $||\mathbf{x}||$ — *норма*, или Евклидова длина вектора $\mathbf{x}$; $\mathbf{x}$ и $\mathbf{u}$ — пара векторов в открытом множестве $\mathbf{M}$ в нормализованном векторном пространстве (состояний). Тогда, согласно условию Липшица, существует такая константа K, что [468], [500]:

$$||\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{u})|| \leq K||\mathbf{x} - \mathbf{u}|| \tag{14.4}$$

для всех $\mathbf{x}$ и $\mathbf{u}$, принадлежащих $\mathbf{M}$. Вектор-функция $\mathbf{F}(\mathbf{x})$, удовлетворяющая условию (14.4), называется функцией Липшица, а K — константой Липшица для $\mathbf{F}(\mathbf{x})$. Неравенство (14.4) также подразумевает непрерывность функции $\mathbf{F}(\mathbf{x})$ по $\mathbf{x}$. Отсюда следует, что в случае автономности системы условие Липшица гарантирует как существование, так и единственность решения уравнения (14.2). В частности, если частные производные $\partial F_i/\partial x_j$ конечны в любой точке, то функция $\mathbf{F}(\mathbf{x})$ удовлетворяет условию Липшица.

Теорема о дивергенции

Рассмотрим область, образованную в пространстве состояний автономной системы телом объемом V и поверхностью S, и обратим внимание на "поток" точек этой области. Из вышесказанного известно, что вектор скорости $d\mathbf{x}/dt$ равен векторному полю $\mathbf{F}(\mathbf{x})$. Предполагая, что векторное поле $\mathbf{F}(\mathbf{x})$ в объеме V "ведет себя хорошо", можно применить теорему о дивергенции (divergence theorem) из векторного исчисления [503]. Пусть $\mathbf{n}$ — единичный вектор, нормальный к поверхности dS и направленный вовне замкнутого объема. Тогда, согласно теореме о дивергенции, должно выполняться соотношение

$$\int_S (\mathbf{F}(\mathbf{x}) \cdot \mathbf{n}) dS = \int_V (\nabla \cdot \mathbf{F}(\mathbf{x})) dV \tag{14.5}$$

между интегралом дивергенции по объему и интегралу по поверхности, направленной вовне нормальной составляющей $\mathbf{F}(\mathbf{x})$. Величину в левой части (14.5) можно понимать как общий *поток*, вытекающий из области, ограниченной замкнутой поверхностью S. Если эта величина равна нулю, система считается консервативной (conservative), если отрицательна —диссипативной (dissipative). В свете выражения (14.5) можно утверждать, что если дивергенция $\nabla \cdot \mathbf{F}(\mathbf{x})$ (скаляр) равна нулю, то система консервативна, а если меньше нуля, то это — диссипативная система.

14.3. Устойчивость состояний равновесия

Рассмотрим автономную динамическую систему, описываемую уравнением в пространстве состояний (14.2). Вектор-константа $\bar{\mathbf{x}} \in \mathbf{M}$ называется равновесным (стационарным) состоянием (equilibrium (stationary) state) системы, если выполняется условие

$$\mathbf{F}(\bar{\mathbf{x}}) = \mathbf{0}, \tag{14.6}$$

где $\mathbf{0}$ — нулевой вектор. Вектор скорости $d\mathbf{x}/\mathrm{dt}$ в точке равновесия исчезает; таким образом, функция-константа $\mathbf{x}(t) = \bar{\mathbf{x}}$ является решением уравнения (14.2). Более того, по причине единственности решения никакая другая кривая решения не проходит через состояние равновесия $\bar{\mathbf{x}}$. Состояние равновесия также называют сингулярной точкой (singular point), отмечая тот факт, что в случае отклонения от точки равновесия траектория снова приведет в эту точку.

Для того чтобы выработать глубокое понимание условия равновесия, предположим, что нелинейная функция $\mathbf{F}(\mathbf{x})$ достаточно гладкая для того, чтобы уравнение (14.2) в пространстве состояний можно было линеаризовать в окрестности точки $\bar{\mathbf{x}}$. Пусть

$$\mathbf{x}(t) = \bar{\mathbf{x}} + \Delta\mathbf{x}(t), \tag{14.7}$$

где $\Delta\mathbf{x}(t)$ — малое отклонение от точки $\bar{\mathbf{x}}$. Тогда, принимая во внимание только первые два слагаемые разложения в ряд Тейлора функции $\mathbf{F}(\mathbf{x})$, можно записать следующую ее аппроксимацию:

$$\mathbf{F}(\mathbf{x}) \simeq \bar{\mathbf{x}} + \mathbf{A}\Delta\mathbf{x}(t). \tag{14.8}$$

Матрица $\mathbf{A}$ является Якобианом нелинейной функции $\mathbf{F}(\mathbf{x})$, вычисленной в точке $\mathbf{x}=\bar{\mathbf{x}}$:

$$\mathbf{A} = \frac{\partial}{\partial \mathbf{x}}\mathbf{F}(\mathbf{x})\bigg|_{\mathbf{x}=\bar{\mathbf{x}}}. \tag{14.9}$$

Подставив (14.7) и (14.8) в (14.2) и воспользовавшись определением состояния равновесия, получим:

$$\frac{\partial}{\partial t}\Delta\mathbf{x}(t) \simeq \mathbf{A}\Delta\mathbf{x}(t). \tag{14.10}$$

ТАБЛИЦА 14.1. Классификация состояний равновесия для систем второго порядка

Тип состояния равновесия $\bar{\mathbf{x}}$	*Собственные значения матрицы Якобиана* $\mathbf{A}$
Устойчивый узел	Вещественные и отрицательные
Устойчивый фокус	Комплексные с отрицательной действительной частью
Неустойчивый узел	Вещественные и положительные
Неустойчивый фокус	Комплексные с положительной действительной частью
Седловая точка	Вещественные с противоположными знаками
Центр	Чисто мнимые

Предполагая, что Якобиан $\mathbf{A}$ несингулярен и, следовательно, обратная матрица $\mathbf{A}^{-1}$ существует, для определения локального поведения траекторий системы в окрестности равновесного состояния $\bar{\mathbf{x}}$ достаточно аппроксимации, описанной формулой (14.10). Если Якобиан $\mathbf{A}$ несингулярен, природа состояния равновесия в основном определяется собственными значениями (eigenvalue) и может быть классифицирована соответствующим образом. В частности, если матрица Якобиана $\mathbf{A}$ имеет m собственных значений с положительными действительными частями, говорят, что равновесное состояние $\bar{\mathbf{x}}$ имеет тип m.

В частном случае для системы второго порядка можно выделить шесть типов состояний равновесия (табл. 14.1 и рис. 14.4) [69], [209]. Без потери общности предполагается, что состояние равновесия достигается в начале координат, т.е. $\bar{\mathbf{x}} = \mathbf{0}$. Обратите внимание, что только в случае седловой точки (saddle point) (см. рис. 14.4, *д*) траектории, входящие в нее, устойчивы, а исходящие из нее — неустойчивы.

Определения устойчивости

Как уже отмечалось, линеаризация уравнения в пространстве состояний дает полезную информацию о свойствах локальной устойчивости (local stability) состояния равновесия. Однако, чтобы иметь возможность более детального исследования устойчивости нелинейных динамических систем, нужны строгие определения устойчивости и сходимости для состояния равновесия.

В контексте автономных нелинейных динамических систем с состоянием равновесия $\bar{\mathbf{x}}$ определения устойчивости и сходимости записываются следующим образом [209].

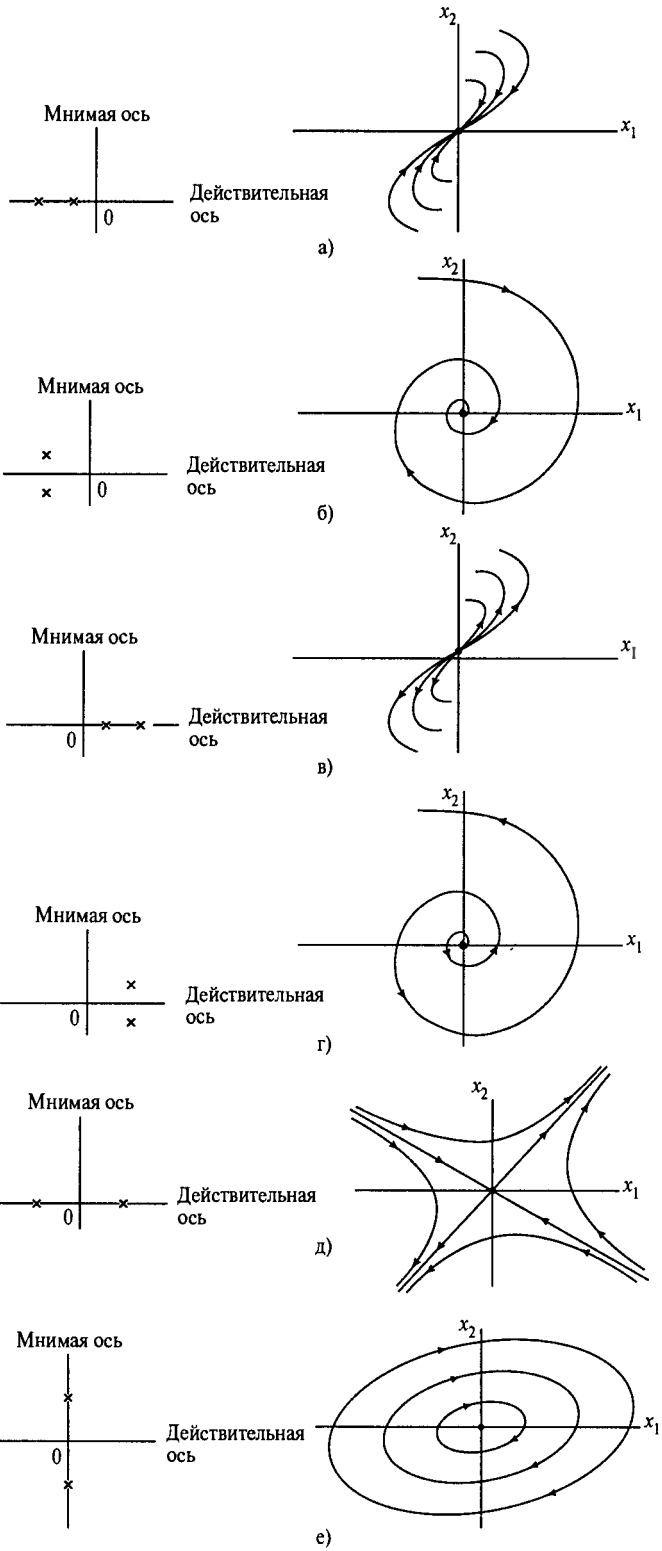


Рис. 14.4. Устойчивый узел (а); устойчивый фокус (б); неустойчивый узел (в); неустойчивый фокус (г); седловая точка (д); цен р (е)

Определение 1. Состояние равновесия $\bar{\mathbf{x}}$ называется равномерно устойчивым, если для любого положительного ε существует такое положительное δ, что из условия

$$||\mathbf{x}(0) - \bar{\mathbf{x}}|| < \delta$$

следует, что

$$||\mathbf{x}(t) - \bar{\mathbf{x}}|| < \varepsilon \text{ для всех } t > 0.$$

Это определение утверждает, что траектория системы может оставаться внутри небольшой окрестности состояния равновесия $\bar{\mathbf{x}}$, если начальное состояние $\mathbf{x}(0)$ близко к равновесному $\bar{\mathbf{x}}$.

Определение 2. Состояние равновесия $\bar{\mathbf{x}}$ называется сходящимся, если существует такое положительное δ, что из условия

$$||\mathbf{x}(0) - \bar{\mathbf{x}}|| < \delta$$

следует, что

$$\mathbf{x}(t) \to \bar{\mathbf{x}} \text{ при } t \to \infty.$$

Значение второго определения состоит в том, что если начальное состояние $\mathbf{x}(0)$ траектории достаточно близко к состоянию равновесия $\bar{\mathbf{x}}$, то траектория, описываемая вектором состояний $\mathbf{x}(t)$, достигает точки $\bar{\mathbf{x}}$ при устремлении времени t к бесконечности.

Определение 3. Равновесное состояние $\bar{\mathbf{x}}$ называется асимптотически устойчивым, если оно одновременно является устойчивым и сходящимся.

Здесь следует заметить, что устойчивость и сходимость являются независимыми свойствами. Только в том случае, когда оба они присутствуют, наблюдается асимптотическая устойчивость.

Определение 4. Состояние равновесия $\bar{\mathbf{x}}$ называется глобально асимптотически устойчивым, если оно устойчиво и все траектории системы сходятся к точке $\bar{\mathbf{x}}$ при устремлении времени t к бесконечности.

Это определение подразумевает, что система не имеет других состояний равновесия и что все траектории системы остаются ограниченными для всех моментов времени $t > 0$. Другими словами, глобальная асимптотическая устойчивость подразумевает, что система в конце концов придет в состояние $\bar{\mathbf{x}}$ при любых начальных условиях.

Пример 14.1

Пусть решение $\mathbf{u}(t)$ нелинейной динамической системы, описываемой уравнением (14.2), изменяется во времени (рис. 14.5). Для того чтобы это решение было равномерно устойчивым, требуется, чтобы $\mathbf{u}(t)$ и любое другое решение $\mathbf{v}(t)$ оставались достаточно близкими друг к другу при одинаковых значениях t (см. рис. 14.5). Такой тип поведения называется изохронным соответствием (isochronous correspondence) двух решений $\mathbf{v}(t)$ и $\mathbf{u}(t)$ [499]. Сходимость решения $\mathbf{u}(t)$ подразумевает, что для любого другого решения $\mathbf{v}(t)$, для которого $||\mathbf{v}(0)-\mathbf{u}(0)|| \leq \delta(\varepsilon)$ в момент времени $t = 0$, решение $\mathbf{v}(t)$ также сходится к состоянию равновесия при устремлении времени t к бесконечности.

■

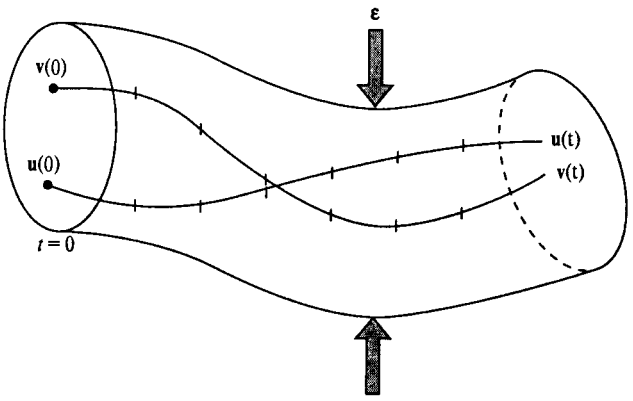


Рис. 14.5. Понятие равномерной устойчивости (сходимости) вектора состояния

Теоремы Ляпунова

Определив устойчивость и асимптотическую устойчивость состояния равновесия в динамических системах, можно приступить к рассмотрению вопроса о том, как определить устойчивость. Естественно, это можно сделать с помощью фактического нахождения всех возможных решений уравнений системы в пространстве состояний, однако такой подход достаточно сложен, а иногда и невозможен. Более элегантный подход можно найти в современной теории устойчивости (modern stability theory), основоположником которой является Ляпунов. В частности, можно исследовать задачу устойчивости, применив прямой метод Ляпунова (direct method of Lyapunov), который использует непрерывную скалярную функцию вектора состояний, называемую функцией Ляпунова.

Теоремы устойчивости и асимптотической устойчивости Ляпунова для уравнения (14.2) в пространстве состояний, описывающего автономную нелинейную динамическую систему с вектором состояний $\mathbf{x}(t)$ и состоянием равновесия $\bar{\mathbf{x}}$, формулируются следующим образом.

Теорема 1. Состояние равновесия $\bar{\mathbf{x}}$ является устойчивым, если в малой окрестности этой точки существует положительно определенная функция $V(\mathbf{x})$, такая, что ее производная по времени в этой области является отрицательно полуопределенной.

Теорема 2. Состояние равновесия $\bar{\mathbf{x}}$ является асимптотически устойчивым, если в малой окрестности этой точки существует положительно определенная функция $V(\mathbf{x})$, такая, что ее производная по времени в этой области является отрицательно определенной.

Скалярная функция $V(\mathbf{x})$, удовлетворяющая этим требованиям, называется функцией Ляпунова для состояния равновесия $\bar{\mathbf{x}}$.

Эти теоремы требуют, чтобы функция Ляпунова $V(\mathbf{x})$ была положительно определенной. Такая функция определяется следующим образом.

Функция $V(\mathbf{x})$ называется положительно определенной в пространстве состояний $\mathbf{G}$, если для всех $\mathbf{x} \in \mathbf{G}$ выполняются следующие условия.

1. Функция $V(\mathbf{x})$ имеет непрерывные частные производные по всем компонентам вектора состояний $\mathbf{x}$.
2. $V(\bar{\mathbf{x}}) = 0$.
3. $V(\mathbf{x}) > 0$ при $\mathbf{x} \neq \bar{\mathbf{x}}$.

Для заданной функции Ляпунова $V(\mathbf{x})$, согласно теореме 1, равновесное состояние $\bar{\mathbf{x}}$ будет устойчивым, если[2]

$$\frac{d}{dt}V(\mathbf{x}) \leq 0 \quad \text{для } \mathbf{x} \in \mathbf{U} - \bar{\mathbf{x}}, \tag{14.11}$$

где $\mathbf{U}$ — малая окрестность точки $\bar{\mathbf{x}}$. Продолжая далее, согласно теореме 2, состояние $\bar{\mathbf{x}}$ будет асимптотически устойчивым, если

$$\frac{d}{dt}V(\mathbf{x}) < 0 \quad \text{для } \mathbf{x} \in \mathbf{U} - \bar{\mathbf{x}}. \tag{14.12}$$

Важной точкой этой дискуссии является то, что теоремы Ляпунова можно применить, не решая само уравнение системы в пространстве состояний. К сожалению, в этих теоремах ничего не говорится о методах нахождения самих функций Ляпунова — это является уделом метода проб, ошибок и озарения, который нельзя обобщить. Во многих задачах функцией Ляпунова может служить функция энергии. Заметим, что невозможность нахождения соответствующей функции Ляпунова не является доказательством неустойчивости системы. Существование функции Ляпунова является достаточным, но не необходимым условием устойчивости системы.

Функция Ляпунова $V(\mathbf{x})$ подводит математический базис под анализ глобальной устойчивости нелинейных динамических систем, описываемых уравнением (14.2). С другой стороны, использование выражения (14.10) основано на матрице Якобиана $\mathbf{A}$. Эта матрица является основой анализа локальной устойчивости системы. Выводы анализа глобальной устойчивости являются более мощными, чем выводы анализа локальной устойчивости. Таким образом, любая глобально устойчивая система является локально устойчивой. Обратное утверждение неверно.

---

[2] В дополнение к равенству (14.11) в общем случае глобальная устойчивость нелинейных динамических систем требует выполнения условия радиальной неограниченности [1002]:

$V(\mathbf{x}) \rightarrow 0$ при $||\mathbf{x}|| \rightarrow \infty$.

Это условие обычно удовлетворяется функцией Ляпунова, построенной для нейронных сетей с сигмоидальной функцией активации.

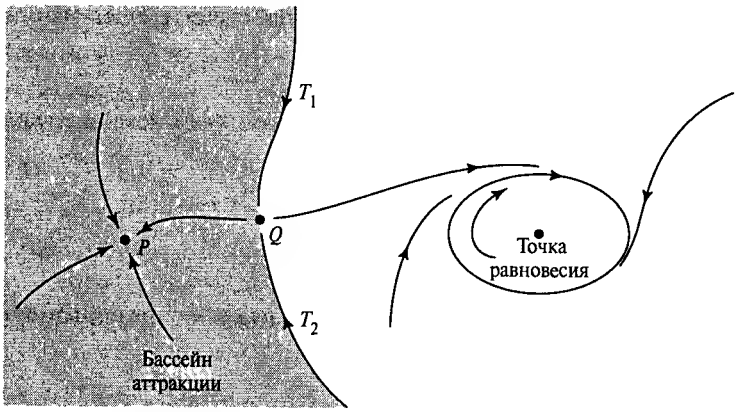


Рис. 14.6. Понятие бассейна аттракции и идеи сепаратрисы

14.4. Аттракторы

Диссипативные системы в общем случае характеризуются наличием множества аттракторов или множествами, имеющим размерность, меньшую размерности пространства состояний. Здесь под "множеством" понимается некоторая k-мерная поверхность, содержащаяся в N-мерном пространстве состояний и описываемая следующей системой уравнений:

$$M_j(x_1, x_2, \ldots, x_N) = 0, \quad \begin{cases} j = 1, 2, \ldots, k, \\ k < N, \end{cases} \tag{14.13}$$

где $x_1, x_2, \ldots, x_N$ — элементы N-мерного вектора состояний системы; M_j — некоторая функция этих элементов. Такие множества называют аттракторами[3], отмечая тот факт, что они являются ограниченными подмножествами, к которым сходятся области начальных состояний пространства состояний ненулевого объема с течением времени t [808].

Это множество может состоять из одной точки в пространстве состояний; в этом случае говорят о точечном аттракторе. Это множество может также принимать форму периодической орбиты; в этом случае говорят об устойчивом предельном цикле (limit cycle). Здесь под устойчивостью понимается асимптотическая сходимость к этому циклу близлежащих траекторий. На рис. 14.6 продемонстрированы эти два типа аттракторов. Аттракторы представляют собой единственные равновесные состояния

[3] В качестве строгого определения аттрактора можно предложить следующее [612], [640]. Подмножество M пространства состоянnй называется аттрактором, еслn:

- M инвариантно к потоку;
- существует некоторая (открытая) окрестность M, которая стягивается к M для заданного потока (flow);
- никакая часть M не является переходной;
- не существует декомпозиции M на две непересекающиеся инвариантные части.

динамической системы, которые можно наблюдать экспериментально. Однако следует заметить, что в контексте аттракторов равновесное состояние не обязательно является статическим или устойчивым. Например, предельный цикл представляет собой устойчивое состояние аттрактора, которое непрерывно изменяется во времени.

На рис. 14.6 видно, что каждый из аттракторов окружен собственной четко очерченной областью. Такая область называется бассейном (областью) аттракции (basin или domain of attraction). Также заметим, что каждое начальное состояние системы находится в бассейне какого-либо аттрактора. Граница, отделяющая один бассейн притяжения от другого, называется сепаратрисой (separatrix). В случае, показанном на рис. 14.6, граница бассейна представлена объединением траекторий T_1 и T_2 и седловой точки Q.

Предельный цикл является типичной формой осцилляции, которая возникает в случае, когда точка равновесия в нелинейной системе становится неустойчивой. Осцилляция может возникнуть в нелинейной системе любого порядка. Тем не менее предельные циклы являются типичной характерной чертой систем второго порядка.

Гиперболические аттракторы

Рассмотрим точечный аттрактор, нелинейные динамические уравнения которого линеаризованы в окрестности равновесного состояния $\bar{\mathbf{x}}$ (см. раздел 14.2). Обозначим символом $\mathbf{A}$ матрицу Якобиана этой системы, вычисленную в точке $\mathbf{x} = \bar{\mathbf{x}}$. Аттрактор называется гиперболическим, если все собственные значения матрицы Якобиана $\mathbf{A}$ имеют абсолютное значение меньше единицы [808]. К примеру, поток гиперболического аттрактора второго порядка может иметь формы, показанные на рис. 14.4, *а*, *б*. В обоих случаях собственные значения матрицы Якобиана $\mathbf{A}$ имеют отрицательные действительные части. Гиперболические аттракторы представляют особый интерес при изучении задачи, известной под названием задачи обращающихся в нуль градиентов (vanishing gradients problem), которая возникает в динамически управляемых рекуррентных сетях. Эта задача будет рассматриваться в следующей главе.

14.5. Нейродинамические модели

Формализовав поведение нелинейных динамических систем, можно приступать к обсуждению отдельных важных вопросов нейродинамики, что и будет сделано в этом и следующем разделах. Следует подчеркнуть, что не существует универсально согласованного определения того, что понимается под нейродинамикой. Вместо этого определим наиболее общие свойства нейродинамических систем, которые будем рассматривать в настоящей главе. В частности, ограничим рассмотрение теми нейродинамическими системами, которые имеют непрерывные переменные состояний и

уравнения движения которых имеют дифференциальные или разностные формы. Рассматриваемые системы обладают четырьмя общими характеристиками [826], [838].

1. *Большое количество степеней свободы.* Кора головного мозга человека представляет собой в высшей мере параллельную и распределенную систему, которая, по некоторым оценкам, содержит более 10 миллиардов нейронов, а каждый нейрон моделируется одной или несколькими переменными состояний. Общепринято считать, что вычислительная мощность и отказоустойчивость такой нейродинамической системы обусловлена ее коллективной динамикой. Эта система характеризуется большим количеством связывающих констант, представленных силой (эффективностью) отдельных синаптических соединений.
2. *Нелинейность.* Нейродинамическая система является нелинейной. Фактически нелинейность существенна при создании универсальных вычислительных машин.
3. *Диссипативность.* Нейродинамическая система является диссипативной (dissipative). Таким образом, она характеризуется сходимостью с течением времени объема пространства состояний к некоторому множеству меньшей размерности.
4. *Шум.* Является существенной характеристикой нейродинамических систем. В реальном нейроне мембранный шум генерируется в синаптических соединениях [545].

Наличие шума обусловливает использование вероятностного толкования деятельности нейрона, что выводит анализ нейродинамических систем на более высокий уровень сложности. Подробное изучение стохастической нейродинамики выходит за рамки настоящей книги, и в дальнейшем мы будем игнорировать влияние шума.

Аддитивная модель

Рассмотрим динамическую модель нейрона без учета шума, показанную на рис. 14.7, математические основы которой рассматривались в главе 13. В физических терминах синаптические веса $w_{j1}, w_{j2}, \ldots, w_{jN}$ представляют собой *емкости*, а соответствующие входные сигналы $x_1(t), x_2(t), \ldots, x_N(t)$ — потенциалы, где N — количество входов. Эти входы подаются на суммирующие соединения, которые характеризуются следующими свойствами.

- Низкое входное сопротивление.
- Единичный ток в цепи.
- Высокое выходное сопротивление.

Таким образом, эти соединения играют роль узлов суммирования входных токов. Общий ток, протекающий в направлении от входного узла нелинейного элемента (см. рис. 14.7), можно выразить следующей формулой:

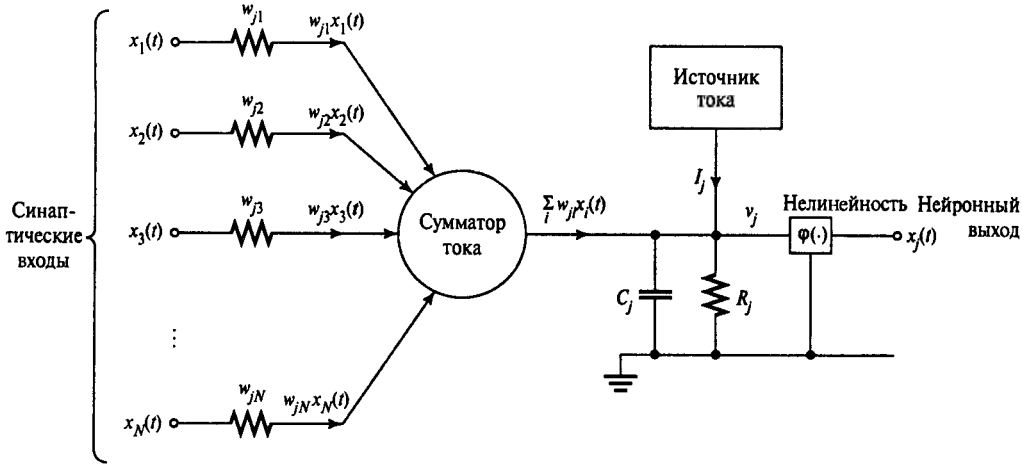


Рис. 14.7. Аддитивная модель нейрона

$$\sum_{i=1}^{N} w_{ji}x_i(t) + I_j,$$

где первое слагаемое (суммирование) отражает возбуждения $x_1(t), x_2(t), \ldots, x_N(t)$, действующие на синаптические веса (емкости) $w_{j1}, w_{j2}, \ldots, w_{jN}$ соответственно; а второе слагаемое — это источник тока I_j, представляющий внешнее смещение. Пусть $v_j(t)$ — индуцированное локальное поле на входе нелинейной функции активации $\varphi(\cdot)$. Тогда общий ток, вытекающий из входного узла нелинейного элемента, можно выразить следующим образом:

$$\frac{v_j(t)}{R_j} + C_j \frac{dv_j(t)}{dt},$$

где первое слагаемое вызвано сопротивлением утечки (leakage resistance) R_j, а второе — емкостью утечки (leakage capacitance) C_j. Из закона Кирхгофа известно, что общий ток, протекающий в сторону любого узла электрической цепи, равен нулю. Применив этот закон к входному узлу нелинейности на рис. 14.7, получим следующее:

$$C_j \frac{dv_j(t)}{dt} + \frac{v_j(t)}{R_j} = \sum_{i=1}^{N} w_{ji}x_i(t) + I_j. \tag{14.14}$$

Емкостное слагаемое $C_j \mathrm{d}v_j(t)/\mathrm{d}t$ в левой части формулы (14.14) является самым простым способом добавления в модель нейрона динамики (памяти). Для заданного индуцированного локального поля $v_j(t)$ можно определить выход нейрона j с

помощью следующего нелинейного выражения:

$$x_j(t) = \varphi(v_j(t)). \tag{14.15}$$

RC-модель, описываемую выражением (14.14), обычно называют аддитивной (additive model). Эта терминология используется для отличия этой модели от мультипликативной (или шунтирующей) модели, в которой w_{ji} зависит от x_i [391].

Характерным признаком аддитивной модели, описываемой выражением (14.14), является то, что сигнал $x_i(t)$, применяемый к нейрону j связанным нейроном i, является медленно изменяющейся функцией времени t. Описанная модель составляет основу классической нейродинамики (classical neurodinamics)[4].

Теперь рассмотрим рекуррентную сеть, состоящую из N взаимосвязанных нейронов, каждый из которых имеет одну и ту же математическую модель, описываемую выражениями (14.14) и (14.15). Тогда, игнорируя запаздывания, связанные с распространением сигнала между нейронами, можно определить динамику такой сети следующей системой дифференциальных уравнений первого порядка:

$$C_j \frac{dv_j(t)}{dt} = -\frac{v_j(t)}{R_j} + \sum_{i=1}^{N} w_{ji} x_i(t) + I_j, \quad j = 1, 2, \ldots, N. \tag{14.16}$$

Эта система имеет ту же математическую форму, что и уравнения состояний (14.1), и получается простой перестановкой слагаемых в равенстве (14.14). Здесь предполагается, что функция активации $\varphi(\cdot)$, определяющая отношение выхода $x_j(t)$

---

[4] **Интегральный нейрон**

Аддитивная модель (14.14) не в полной мере отражает сущность биологнческого нейрона. В частности, она игнорирует временную информацию, закодированную в потенциале действия (потенциалы действия кратко описаны в качественных терминах в главе 1). В [476] описывается динамическая модель, которая учитывает потенциалы действия, при этом рассматривалась интегральная модель нейрона. Работа такого нейрона описывается дифференциальным уравнением первого порядка:

$$C\frac{d}{dt}u(t) = -\frac{1}{R}(u(t) - u_0) + i(t), \tag{1}$$

где $u(t)$ — внутренний потенциал нейрона; — емкость мембранного окружения нейрона; R — сопротивление утечки в мембране; $i(t)$ — электрический ток, протекающий в нейрон из другого нейрона; u_0 — потенциал, к которому приходит нейрон, когда $i(t)$ исчезает.

Потенциал действия генерируется каждый раз, когда внутренний потенциал $u(t)$ достигает значения порога.

Потенцнал действия рассматривается как дельта-функция Дирака:

$$g_k(t) = \sum_n \delta(t - t_{k,n}), \tag{2}$$

где $t_{k,n}, n = 1, 2, \ldots$ — моменты времени, в которые нейрон k "выстреливает" потенциал действия. Эти моменты времени определяются уравнением (1).

Общий ток $\mathrm{i}_k(t)$, втекающий в нейрон k, моделируется следующим образом:

$$\frac{d}{dt}i_k(t) = -\frac{1}{\tau}i_k(t) + \sum_j w_{kj} g_j(t), \tag{3}$$

где w_{kj} — синаптический вес, идущий от нейрона j к нейрону k, а функция $g_j(t)$ определяется по формуле (2).

Аддитивную модель в выражении (14.14) можно рассматривать как частный случай модели (3). В частности, игольчатую природу $g_j(t)$ можно игнорировать, заменив эту функцию ее же сверткой с некоторой гладкой функцией. Такая замена вполне обоснована, если в течение разумного интервала времени в связи с высокой связностью вклад в сумму в правой части выражения (3) велик, и все, что нас интересует — это краткосрочное поведение уровня выходного "выстрела" нейрона k.

нейрона j к его же индуцированному локальному полю $v_j(t)$, является непрерывной и, следовательно, дифференцируемой. Чаще всего в качестве функции активации используют логистическую функцию

$$\varphi(v_j) = \frac{1}{1+\exp(-v_j)}, \quad j = 1, 2, \ldots, N. \tag{14.17}$$

Необходимым условием для существования алгоритмов, описываемых в разделах 14.6–14.11, является наличие в рекуррентных сетях, описываемых уравнениями (14.15) и (14.16), фиксированных точек (т.е. точечных аттракторов).

Связанная модель

Для упрощения выкладок предположим, что константа времени $\tau_j = R_j C_j$ нейрона j в (14.16) является одинаковой для всех j. Тогда, нормализуя время t по отношению к общему значению этой константы и нормализуя w_{ji} и I_j по отношению к R_j, модель (14.16) можно переписать в виде

$$\frac{dv_j(t)}{dt} = -v_j(t) + \sum_i w_{ji}\varphi(v_i(t)) + I_j, \quad j = 1, 2, \ldots, N, \tag{14.18}$$

где учитывается уравнение (14.15). Структура аттрактора системы нелинейных дифференциальных уравнений первого порядка (14.18) является в сущности такой же, как и в тесно связанной модели, описываемой следующим образом [840]:

$$\frac{dx_j(t)}{dt} = -x_j(t) + \varphi\left(\sum_i w_{ji}x_i(t)\right) + K_j, \quad j = 1, 2, \ldots, N. \tag{14.19}$$

В аддитивной модели (14.18) индуцированные локальные поля $v_1(t), v_2(t), \ldots, v_N(t)$ отдельных нейронов образуют вектор состояний. С другой стороны, в связанной модели (14.19) вектор состояний образуют выходы нейронов $x_1(t), x_2(t), \ldots, x_N(t)$.

Эти две нейродинамические модели связаны друг с другом линейным обратимым преобразованием. В частности, умножив обе стороны уравнения (14.19) на w_{kj}, выполнив суммирование по j и подставив преобразование

$$v_k(t) = \sum_j w_{kj}x_j(t),$$

получим модель, описываемую выражением (14.18), и, таким образом, установим, что слагаемые смещения в этих двух моделях связаны соотношением

$$I_k = \sum_j w_{kj} K_j.$$

Здесь важно отметить, что результаты, полученные для устойчивости аддитивной модели (14.18), применимы и к связанной модели (14.19).

Чтобы покончить с вопросом отношений между двумя описанными нейродинамическими моделями, на рис. 14.8 они показаны в виде блочных диаграмм; рис. 14.8, *а, б* соответствуют матричной записи выражений (14.18) и (14.19). Здесь $\mathbf{W}$ — матрица синаптических весов; $\mathbf{v}(t)$ — вектор индуцированных полей в момент времени t; $\mathbf{x}(t)$ — вектор выходов нейронов в момент времени t. В обеих моделях на рис. 14.8 явно выражена обратная связь.

14.6. Управление аттракторами как парадигма рекуррентных сетей

Когда количество нейронов N в нейродинамической модели, описываемой уравнением (14.16), достаточно велико, она обладает всеми общими свойствами, перечисленными в разделе 14.5 (за исключением влияния шума): большим числом степеней свободы, нелинейностью и диссипативностью. Следовательно, такая нейродинамическая модель может иметь сложную структуру аттракторов и, таким образом, обладать полезными вычислительными возможностями.

Определение аттракторов для вычислительных объектов (например, ассоциативной памяти, операторов отображения входа на выход) является одной из основных парадигм нейронных сетей. Для того чтобы реализовать эту идею, требуется управлять местом размещения аттракторов в пространстве состояний системы. При этом алгоритм обучения принимает форму нелинейного динамического уравнения, которое управляет расположением аттракторов для кодирования информации в желаемом виде или обучения рассматриваемых временных структур. Действуя таким способом, можно установить тесную взаимосвязь между физикой и алгоритмами вычислений.

Одним из способов, которым коллективные свойства нейронных сетей могут использоваться для реализации вычислительных задач, является применение концепции минимизации энергии. Сеть Хопфилда и модель состояния мозга представляют собой примеры ассоциативной памяти, не содержащей скрытые нейроны. Ассоциативная память является важным ресурсом для интеллектуализации поведения. Еще одной нейродинамической моделью является модель оператора отображения входа на выход (input-output mapper), работа которого предполагает наличие скрытых нейронов. В последнем случае для минимизации функции стоимости, определенной в терминах

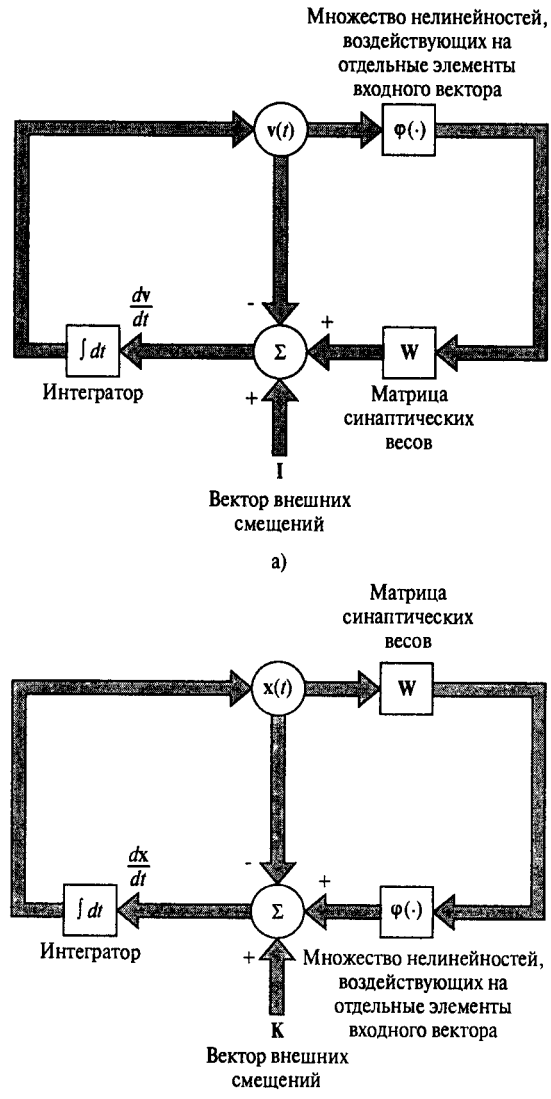


Рис. 14.8. Блочная диаграмма нейродинамической системы, описываемой дифференциальными уравнениями первого порядка (14.18) (а); блочная диаграмма модели, описываемой уравнением (14.19) (б)

параметров нейронной сети, и, таким образом, для изменения положения аттракторов часто используется метод наискорейшего спуска. Последнее приложение нейродинамической модели показано на примере динамически управляемой рекуррентной сети, которую мы рассмотрим в следующей главе.

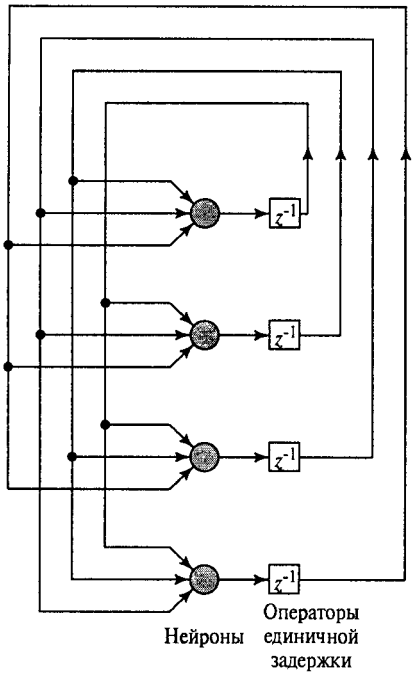


Рис. 14.9. Архитектурный граф сети Хопфилда, содержащей $N = 4$ нейрона

14.7. Модель Хопфилда

Сеть (модель) Хопфилда состоит из множества нейронов и соответствующего множества единичных задержек, формирующих систему с множеством обратных связей (multiple-loop feedback system) (рис. 14.9). Количество обратных связей равно количеству нейронов. Как правило, выход каждого нейрона замыкается через элемент единичной задержки на все остальные нейроны сети. Другими словами, нейрон этой сети не имеет обратных связей с самим собой. Причина исключения из рассмотрения собственных обратных связей будет рассмотрена немного позже.

Для изучения динамики сетей Хопфилда будет использоваться нейродинамическая модель, описываемая уравнением (14.16), которая основана на аддитивной модели нейрона.

Вспоминая, что $x_i(t) = \varphi_i(v_i(t))$, уравнение (14.16) можно переписать в следующем виде:

$$C_j \frac{d}{dt} v_j(t) = -\frac{v_j(t)}{R_j} + \sum_{i=1}^{N} w_{ji} \varphi_i(v_i(t)) + I_j, \quad j = 1, 2, \ldots, N. \tag{14.20}$$

Чтобы продолжить дискуссию, сделаем следующие допущения.

1. Матрица синаптических весов является симметричной:

$$w_{ji} = w_{ij} \text{ для всех } i \text{ и } j. \quad (14.21)$$

2. Каждый нейрон имеет собственную нелинейную функцию активации (поэтому в записи $\varphi_i(\cdot)$ в выражении (14.20) присутствует индекс i).
3. Для всех нелинейных функций активации существуют *обратные*, т.е. можно записать:

$$v = \varphi_i^{-1}(x). \quad (14.22)$$

Пусть сигмоидальная функция $\varphi_i(v)$ определяется как гиперболический тангенс:

$$x = \varphi_i(v) = \tanh\left(\frac{a_i v}{2}\right) = \frac{1 - \exp(-a_i v)}{1 + \exp(-a_i v)}, \quad (14.23)$$

который в начале координат имеет наклон, равный $a_i/2$:

$$\frac{a_i}{2} = \left.\frac{d\varphi_i}{dv}\right|_{v=0}. \quad (14.24)$$

Исходя из этого, a_i называется коэффициентом передачи (gain) нейрона i.

Обратное преобразование выхода во вход (14.22) можно переписать в следующем виде:

$$v = \varphi_i^{-1}(x) = -\frac{1}{a_i}\log\left(\frac{1-x}{1+x}\right). \quad (14.25)$$

Стандартная форма этого обратного соотношения для нейрона с единичным коэффициентом передачи будет иметь следующий вид:

$$\varphi^{-1}(x) = -\log\left(\frac{1-x}{1+x}\right). \quad (14.26)$$

Выражение (14.25) можно переписать в терминах стандартного соотношения:

$$\varphi_i^{-1}(x) = \frac{1}{a_i}\varphi^{-1}(x). \quad (14.27)$$

На рис. 14.10, *а* показан график стандартной сигмоидальной нелинейности $\varphi(v)$, а на рис. 14.10, *б* — график соответствующей обратной нелнейности $\varphi^{-1}(x)$.

Функция энергии (Ляпунова) сети Хопфилда (см. рис. 14.9) определяется следующим образом [479]:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ji}x_i x_j + \sum_{j=1}^{N}\frac{1}{R_j}\int_0^{x_j}\varphi_j^{-1}(x)dx - \sum_{j=1}^{N} I_j x_j. \qquad (14.28)$$

Функция энергии (14.28) может иметь сложную поверхность с множеством минимумов. Динамика этой сети описывается механизмом, обеспечивающим нахождение этих минимумов.

Исходя из этого, дифференцируя функцию энергии E по времени, получим:

$$\frac{dE}{dt} = -\sum_{j=1}^{N}\left(\sum_{i=1}^{N} w_{ji}x_i - \frac{v_j}{R_j} + I_j\right)\frac{dx_j}{dt}. \qquad (14.29)$$

Из нейродинамического соотношения (14.20) видно, что величина, заключенная в скобки в правой части (14.29), равна $C_j \mathrm{d}v_j/\mathrm{d}t$. Таким образом, можно упростить выражение (14.29) до следующего:

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j\left(\frac{dv_j}{dt}\right)\frac{dx_j}{dt}. \qquad (14.30)$$

Итак, получено обратное соотношение, определяющее v_j в терминах x_j. Подставив (14.22) в (14.30), получим:

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j\left(\frac{d}{dt}\varphi_j^{-1}(x_j)\right)\frac{dx_j}{dt} = -\sum_{j=1}^{N} C_j\left(\frac{dx_j}{dt}\right)^2\left(\frac{d}{dx_j}\varphi_j^{-1}(x_j)\right). \qquad (14.31)$$

На рис. 14.10, *б* видно, что обратное отображение $\varphi_j^{-1}(x_j)$ является монотонно возрастающей функцией выхода x_j. Отсюда следует, что

$$\frac{d}{dx_j}\varphi_j^{-1}(x_j) \geq 0 \quad \text{для всех} \quad x_j. \qquad (14.32)$$

Обратите внимание, что выполняется также и следующее неравенство:

$$\left(\frac{dx_j}{dt}\right)^2 \geq 0 \quad \text{для всех} \quad x_j. \qquad (14.33)$$

Следовательно, все составляющие суммы в правой части (14.31) являются неотрицательными. Другими словами, функция энергии E, определяемая формулой (14.28),

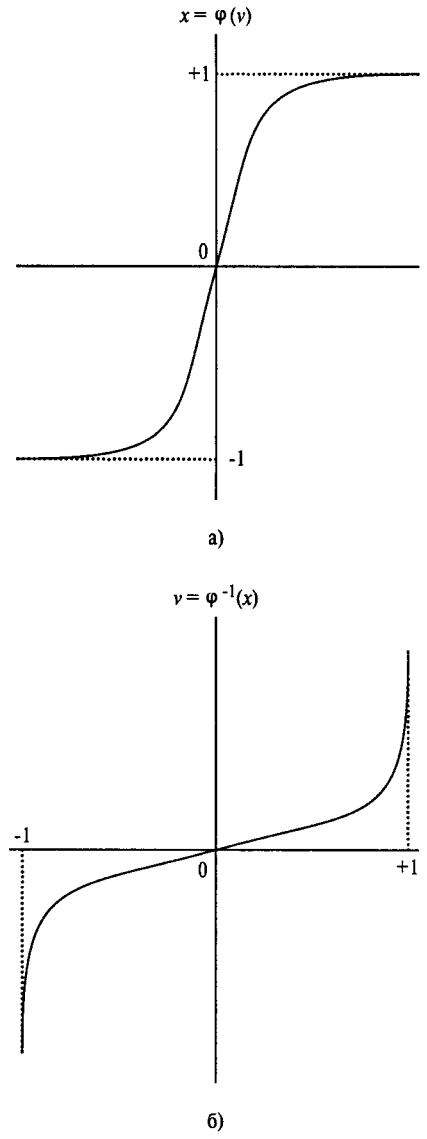


Рис. 14.10. Графики стандартной сигмоидальной нелинейности (а) и обратной к ней функции (б)

удовлетворяет неравенству

$$\frac{dE}{dt} \leq 0.$$

Из определения (14.28) видно, что функция E является ограниченной, и мы можем вывести два следующих утверждения.

1. Функция энергии E является функцией Ляпунова непрерывной модели Хопфилда.
2. Эта модель, согласно теореме 1 Ляпунова, является устойчивой.

Другими словами, эволюция во времени непрерывной модели Хопфилда, описываемой системой нелинейных дифференциальных уравнений первого порядка (14.20), представляет собой некоторую траекторию в пространстве состояний, обеспечивающую нахождение минимума функции энергии (Ляпунова) и приходящую к некоторой фиксированной точке. В выражении (14.31) можно заметить, что производная dE/dt исчезает только тогда, когда

$$\frac{d}{dt}x_j(t) = 0 \quad \text{для всех} \quad j.$$

Таким образом, можно продвинуться еще на один шаг и записать:

$$\frac{dE}{dt} < 0 \text{ за исключением фиксированной точки.} \tag{14.34}$$

Уравнение (14.34) создает предпосылки для формулировки следующей теоремы.

Функция энергии (Ляпунова) сети Хопфилда является монотонно убывающей функцией времени.

Следовательно, сеть Хопфилда является глобально асимптотически устойчивой. Фиксированные точки аттракторов являются точками минимума функции энергии, и наоборот.

Соотношение между устойчивыми состояниями дискретной и непрерывной версии модели Хопфилда

Сеть Хопфилда может работать как в непрерывном, так и в дискретном времени, в зависимости от того, какая модель использовалась для описания нейронов. Непрерывный режим работы, как уже говорилось ранее, основывается на аддитивной модели; дискретный режим работы основан на модели Мак-Каллока–Питца. Можно вывести соотношение между устойчивыми состояниями дискретной и непрерывной моделей Хопфилда, переопределив соотношение между входом и выходом нейрона таким образом, чтобы истинными были две следующие упрощенные характеристики.

1. Выход нейрона имеет следующие асимптотические значения:

$$x_j = \begin{cases} +1 \text{ для } v_j = \infty, \\ -1 \text{ для } v_j = -\infty. \end{cases} \tag{14.35}$$

2. Средняя точка функции активации нейрона находится в начале координат, т.е.

$$\varphi_j(0) = 0. \tag{14.36}$$

Следовательно, можно установить для всех j смещение I_j, равное нулю.

При формулировке функции энергии E для непрерывной модели Хопфилда допускалось наличие у нейрона обратных связей с самим собой. С другой стороны, дискретная модель Хопфилда требует отсутствия таких обратных связей. Чтобы упростить рассмотрение данного вопроса, положим, что в обеих моделях $w_{jj} = 0$ для всех j.

В свете этих наблюдений можно переопределить функцию энергии непрерывной модели Хопфилда (14.28) следующим образом:

$$E = -\frac{1}{2}\sum_{i=1,i\neq j}^{N}\sum_{j=1}^{N} w_{ji}x_i x_j + \sum_{j=1}^{N}\frac{1}{R_j}\int_0^{x_j}\varphi_j^{-1}(x)dx. \tag{14.37}$$

Обратная функция $\varphi_j^{-1}(x)$ определяется формулой (14.27). Таким образом, формулу (14.37) можно переписать в следующем виде:

$$E = -\frac{1}{2}\sum_{i=1,i\neq j}^{N}\sum_{j=1}^{N} w_{ji}x_i x_j + \sum_{j=1}^{N}\frac{1}{a_j R_j}\int_0^{x_j}\varphi^{-1}(x)dx. \tag{14.38}$$

График стандартной формы интеграла

$$\int_0^{x_j}\varphi^{-1}(x)dx$$

показан на рис. 14.11. Он равен нулю в точке x_j =0, и положителен во всех остальных точках. При достижении x_j значений ± 1 значения этого интеграла становятся очень большими. Если же коэффициент передачи a_j (gain) нейрона j становится бесконечно большим (т.е. сигмоидальная нелинейность достигает идеализированной формы с жесткими пределами), второе слагаемое в формуле (14.38) становится настолько малым, что им можно пренебречь. В предельном случае, когда $a_j = \infty$ для всех j, максимумы и минимумы непрерывной модели Хопфилда становятся идентичными максимумам и минимумам дискретной модели. В последнем случае функция энергии (Ляпунова) определяется следующим образом:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1,i\neq j}^{N} w_{ji}x_i x_j, \tag{14.39}$$

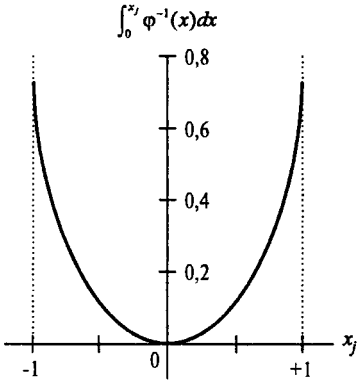


Рис. 14.11. График интеграла $\int_0^{x_j} \varphi^{-1}(x)dx$

где состояние j-го нейрона $x_j = \pm 1$. Таким образом, можно сделать вывод, что только устойчивые точки непрерывной детерминированной модели Хопфилда с очень высоким коэффициентом передачи соответствуют устойчивым точкам дискретной стохастической модели Хопфилда.

Однако если все нейроны j имеют большой, но конечный коэффициент передачи a_j, оказывается, что второе слагаемое в правой части (14.38) вносит заметный вклад в функцию энергии непрерывной модели. В частности, этот вклад является большим и положительным вблизи всех граней, границ и углов единичного гиперкуба, определяющего пространство состояний этой модели. С другой стороны, вклад второго слагаемого становится пренебрежимо малым в точках, удаленных от граней этого куба. Следовательно, функция энергии такой модели имеет свои максимумы в углах, а минимумы смещены во внутреннюю область единичного гиперкуба [479].

На рис. 14.12 показана контурная карта энергии (energy contour map) или ландшафт энергии (energy landscape) непрерывной модели Хопфилда, состоящей из двух нейронов. Выходы этих двух нейронов определяют две оси этой карты. Нижний левый и верхний правый углы на рис. 14.12 представляют собой устойчивые минимумы для случая бесконечного коэффициента передачи. Минимумы для случая конечного коэффициента передачи смещаются внутрь куба. Поток к фиксированным точкам (т.е. точкам устойчивого минимума) может интерпретироваться как решение задачи минимизации функции энергии E, определяемой формулой (14.28).

Дискретная модель Хопфилда как ассоциативная память

Сетям Хопфилда в литературе, посвященной ассоциативной или контентно-адресуемой памяти (content-adressable memory), уделяется довольно большое внимание. В этой модели фиксированные точки соответствуют сохраняемым образам. Однако синаптические веса сети, которые обеспечивают нахождение таких фиксированных точек, неизвестны, и задача заключается в их определении. Основной функ-

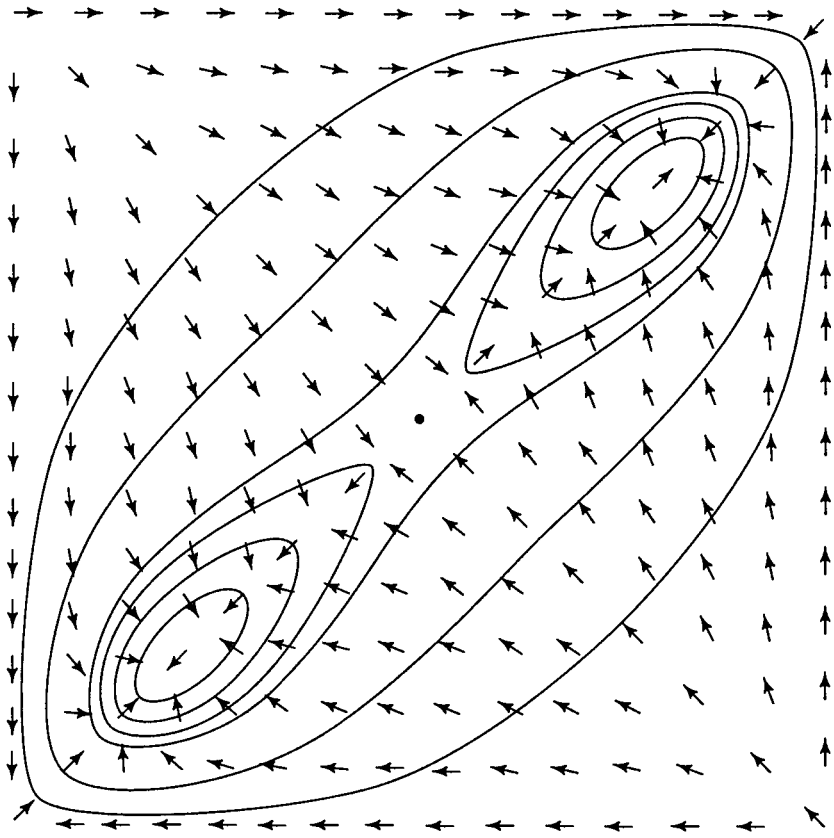


Рис. 14.12. Контурная карта энергии системы с двумя нейронами и двумя устойчивыми состояниями. Ординаты и абсциссы являются выходами двух нейронов. Устойчивые состояния расположены около нижнего левого и верхнего правого углов. Неустойчивые экстремумы расположены в остальных двух углах. Стрелки показывают перемещение состояний. В общем случае эти перемещения не перпендикулярны контурам энергии. (Рисунок взят из [479] с разрешения Национальной академии наук США.)

цией ассоциативной памяти является восстановление образа (объекта), сохраненного в памяти, в ответ на представление неполной или зашумленной версии этого же объекта. Для иллюстрации значения этого утверждения нет ничего лучше, чем привести цитату из [480].

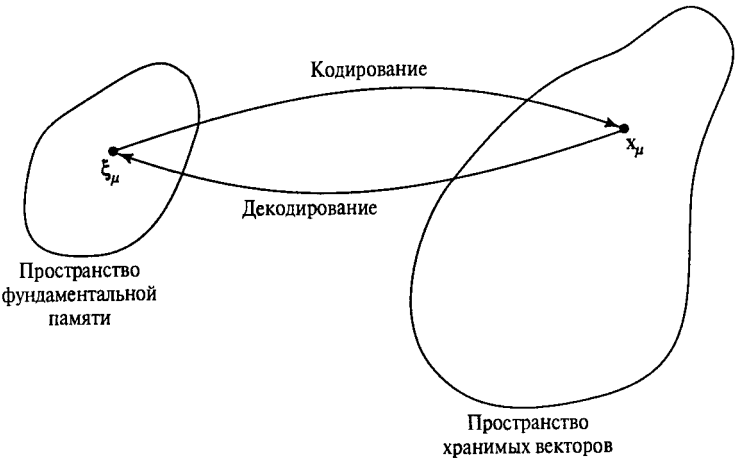


Рис. 14.13. Кодирование-декодирование, выполняемое рекуррентной сетью

"Предположим, что в памяти существует запись "H.A. Kramers & G.H. Wannier Physi Rev. 60, 252 (1941)". Ассоциативная память общего вида будет способна вернуть всю эту запись на основе подачи частичной информации. Ввода строки "& Wannier (1941)" должно быть достаточно. Идеальная память может работать даже при наличии помех и смогла бы вернуть эту запись даже на основе ввода "Wannier (1941)".

Таким образом, важным свойством ассоциативной памяти является восстановление сохраненного образа при подаче разумного подмножества информации этой записи. Более того, ассоциативная память позволяет корректировать ошибки (error-correcting), а именно аннулировать несвязную информацию, если таковая содержится в представленном ей образе.

Сущностью ассоциативной памяти является отображение фундаментальной памяти $\boldsymbol{\xi}_\mu$ в фиксированную (устойчивую) точку $\mathbf{x}_\mu$ некоторой динамической системы (рис. 14.13). Математически это отображение можно представить в следующем виде:

$$\boldsymbol{\xi}_\mu \rightleftarrows \mathbf{x}_\mu.$$

Стрелка, направленная слева направо, описывает операцию кодирования, а стрелка, направленная справа налево, — операцию декодирования. Фиксированные точки аттракторов в пространстве состояний сети являются ячейками фундаментальной памяти (fundamental memories) или состояниями прототипов (prototype state) сети.

Теперь предположим, что сети представлен образ, содержащий частичную, но существенную информацию об одной из ячеек фундаментальной памяти. Тогда этот частичный образ можно представить как начальную точку в пространстве состояний. В принципе, предполагая, что эта стартовая точка находится достаточно близко к

точке, представляющей запрашиваемый образ (т.е. стартовая точка лежит в бассейне аттракции, принадлежащем этой фиксированной точке), система может развиваться во времени и в конце концов прийти к состоянию самой ячейки памяти. В этой точке сетью будет сгенерирован весь образ ячейки памяти. Следовательно, сеть Хопфилда обладает свойством эмерджентности (emergent), что помогает ей извлекать информацию и обрабатывать ошибки.

Если модель Хопфилда использует формальный нейрон [714] и его основной элемент обработки, то каждый такой нейрон имеет два состояния, определяемые уровнем индуцированного локального поля, воздействующего на него. "Включенное" состояние нейрона i соответствует выходу $x_i = +1$, а "выключенное" — выходу $x_i = -1$. Для сети, состоящей из N нейронов, состояние определяется вектором

$$\mathbf{x} = [x_1, x_2, \ldots, x_N]^T.$$

При $x_i = \pm 1$ состояние i представляет один информации, а вектор состояния $\mathbf{x}$ размерности $N \times 1$ представляет двоичное *слово*, содержащее N битов информации.

Индуцированное локальное поле v_i нейрона j определяется следующим образом:

$$v_j = \sum_{i=1}^{N} w_{ji} x_i + b_j, \tag{14.40}$$

где b_j — фиксированное *смещение* (bias), применяемое внешним образом к нейрону j. Таким образом, нейрон j модифицирует свое состояние x_j в соответствии с детерминированным правилом:

$$x_j = \begin{cases} +1, & \text{если } v_j > 0, \\ -1, & \text{если } v_j < 0. \end{cases}$$

Это соотношение может быть переписано в более компактном виде:

$$x_j = \operatorname{sgn}[v_j],$$

где sgn — функция определения знака числа или сигнум-функция (signum function). А что происходит, когда v_j равно нулю? В этом случае действия могут быть произвольными. Например, можно допустить, чтобы при $v_j = 0$ $x_j = \pm 1$. Однако будем использовать следующее соглашение: если $v_j = 0$, нейрон j остается в своем текущем состоянии, независимо от того, включен он или выключен. Значение этого допущения состоит в том, что полученная диаграмма потоков будет симметричной, что и будет продемонстрировано позже.

Дискретные сети Хопфилда, выступающие в качестве ассоциативной памяти, имеют две фазы работы: фазу сохранения и фазу извлечения. Эти фазы подробно описываются ниже.

1. **Фаза сохранения** (storage phase). Предположим, что необходимо сохранить множество N-мерных векторов (двоичных слов), которые обозначим как $\{\boldsymbol{\xi}_\mu | \mu = 1, 2, \ldots, M\}$. Назовем эти векторы ячейками фундаментальной памяти, указывая на то, что в них сохраняются образы, представленные сети. Пусть $\xi_{\mu,i}$ — i-й элемент фундаментальной памяти $\boldsymbol{\xi}_\mu$, где класс $\mu = 1, 2, \ldots, M$. Согласно правилу внешнего произведения векторов (outer product rule), которое является обобщением постулата обучения Хебба, синаптический вес, направленный от нейрона i к нейрону j, определяется так:

$$w_{ji} = \frac{1}{N} \sum_{\mu=1}^{M} \xi_{\mu,j} \xi_{\mu,i}. \tag{14.41}$$

Причиной использования в качестве константы пропорциональности величины $1/N$ является упрощение математического описания процесса извлечения информации. Также заметим, что правило обучения (14.41) представляет собой одношаговую операцию. При нормальной работе сети Хопфилда

$$w_{ii} = 0 \text{ для всех } i. \tag{14.42}$$

Это значит, что нейроны *не имеют* обратных связей с самими собой. Пусть $\mathbf{W}$ — матрица синаптических весов размерности $N \times N$, ji-м элементом которой является синаптический вес w_{ji}. Тогда можно объединить выражения (14.41) и (14.42) в единое уравнение в матричной форме:

$$\mathbf{W} = \frac{1}{N} \sum_{\mu=1}^{M} \boldsymbol{\xi}_\mu \boldsymbol{\xi}_\mu^T - M\mathbf{I}, \tag{14.43}$$

где $\boldsymbol{\xi}_\mu \boldsymbol{\xi}_\mu^T$ — векторное произведение вектора $\boldsymbol{\xi}_\mu$ самого на себя; $\mathbf{I}$ — единичная матрица. Из представленных выше формул-определений синаптических весов (матрицы весов) можем еще раз подтвердить следующее.

1. Выход каждого из нейронов замкнут на входы всех остальных нейронов.
2. В сети отсутствуют обратные связи нейронов с самими собой.
3. Матрица весов этой сети является симметричной, т.е.

$$\mathbf{W}^T = \mathbf{W}. \tag{14.44}$$

Фаза извлечения. Во время фазы извлечения n-мерный вектор $\boldsymbol{\xi}_{\text{проб.}}$, называемый *зондом* (probe), подается сети Хопфилда в качестве ее начального состояния. Элементы этого вектора имеют значения ± 1. Этот вектор обычно представляет собой неполную или зашумленную версию фундаментальной памяти сети. После этого извлечение информации проходит в соответствии с некоторым динамическим правилом, в котором все нейроны j сети *случайно*, но на некотором фиксированном уровне, проверяют индуцированное локальное поле v_j (включающее в себя некоторое внешнее смещение b_j), примененное к ним. Если в некоторый момент времени v_j окажется больше нуля, то нейрон j будет переведен в состояние +1 (если он уже не находится в этом состоянии). Аналогично, если в некоторый момент времени v_j окажется меньше нуля, то нейрон переключится в состояние -1 (если он уже не находится в этом состоянии). Если же v_j будет равно нулю, нейрон j останется в своем текущем состоянии, независимо от того, включенное оно или отключенное. Таким образом, коррекция состояний на каждом шаге будет детерминированной, однако при этом выбор нейрона для осуществления коррекции будет производиться случайным образом. Описанная здесь асинхронная (последовательная) процедура коррекции продолжает выполняться до тех пор, пока изменения не перестанут происходить. Это значит, что, начиная с произвольного пробного вектора $\mathbf{x}$, сеть в конце концов создаст инвариантный во времени вектор состояний $\mathbf{y}$, отдельные элементы которого будут удовлетворять условию устойчивости:

$$y_i = \operatorname{sgn}\left(\sum_{i=1}^{N} w_{ji} y_i + b_j\right), \quad j = 1, 2, \ldots, N, \tag{14.45}$$

или, в матричной форме:

$$\mathbf{y} = \operatorname{sgn}(\mathbf{W}\mathbf{y} + \mathbf{b}), \tag{14.46}$$

где $\mathbf{W}$ — матрица синаптических весов сети; $\mathbf{b}$ — внешний вектор смещения (bias vector). Описанное здесь условие устойчивости также называется условием выравнивания (alignment condition). Удовлетворяющий ему вектор состояния $\mathbf{y}$ называют устойчивым состоянием (stable state), или фиксированной точкой (fixed point) пространства состояний системы. Таким образом, можно сформулировать утверждение о том, что сеть Хопфилда всегда сходится к некоторому устойчивому состоянию, если операция извлечения информации выполняется асинхронно[5].

---

[5] Модель Литтла [660], [662] использует те же синаптические веса, что и модель Хопфилда. Однако их отличие состонт в том, что модель Хопфилда использует *асинхронную динамику*, в то время как модель Литтла — *синхронную*. В результате этн модели показывают различные характеристики сходимости [162], [367]: сеть Хопфилда всегда сходится к устойчивому состоянию, а модель Литтла — либо к устойчивому состоянию, либо к предельному циклу длины не больше двух (это значит, что эти циклы в пространстве состояний сети имеют длину меньшую или равную двум).

ТАБЛИЦА 14.2. Модель Хопфилда

1. *Обучение.* Пусть $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_M$ — известное множество N-мерных ячеек фундаментальной памяти. Используя правило векторного произведения (т.е. постулат обучения Хебба), вычислим синаптические веса сети:

$$w_{ji} = \begin{cases} \frac{1}{N} \sum_{\mu=1}^{M} \xi_{\mu,j} \xi_{\mu,i}, & j \neq i, \\ 0, & j = i, \end{cases}$$

где w_{ji} — синаптический вес, направленный от нейрона i к нейрону j. Элементы вектора $\boldsymbol{\xi}_\mu$ равны ± 1. После того как веса вычислены, они сохраняются в этих фиксированных значениях

2. Инициализация. Пусть $\boldsymbol{\xi}_{\text{проб.}}$ — неизвестный N-мерный входной вектор, представленный сети. Алгоритм инициализируется следующими значениями:

$$x_j(0) = \xi_{j,\text{проб.}}, j = 1, 2, \ldots, N,$$

где $x_j(0)$ — состояние нейрона j в момент времени $n = 0$; $\xi_{j,\text{проб.}}$ — j-й элемент вектора испытаний $\boldsymbol{\xi}_{\text{проб.}}$

3. Итерации до полной сходимости. Выполним корректировку элементов вектора состояний $\mathbf{x}(n)$ в асинхронном режиме (т.е. случайно и по одному элементу за итерацию) в соответствии с правилом

$$x_j(n+1) = \operatorname{sgn}\left(\sum_{i=1}^{N} w_{ji} x_j(n) \right), \quad j = 1, 2, \ldots, N.$$

Повторяем итерации до тех пор, пока вектор состояний $\mathbf{x}$ не перестанет изменяться

4. Формирование выхода. Пусть $\mathbf{x}_{\text{фикс.}}$ — фиксированная точка (устойчивое состояние), вычисленная в результате завершения шага 3. Тогда результирующий выходной вектор будет следующим:

$$\mathbf{y} = \mathbf{x}_{\text{фикс.}}$$

Шаг 1 является фазой запоминания, а шаги 2–4 — фазой извлечения

В табл. 14.2 подытожены действия, составляющие фазы сохранения и извлечения в работе сети Хопфилда.

Пример 14.2

Для того чтобы проиллюстрировать эмерджентное поведение модели Хопфилда, рассмотрим сеть, состоящую из трех нейронов (рис. 14.14, *а*). Эта сеть имеет следующую матрицу весов:

$$\mathbf{W} = \frac{1}{3} \begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}.$$

Данная матрица является допустимой, так как она удовлетворяет условиям (14.42) и (14.44). Предполагается, что внешнее смещение, применяемое ко всем нейронам, равно нулю. Учитывая, что в нашей сети имеются три нейрона, следует рассмотреть $2^3 = 8$ возможных состояний. Из этих восьми состояний только два ($(1, -1, 1)$ и $(-1, 1, -1)$) являются устойчивыми; все оставшиеся шесть состояний неустойчивы. Речь идет об устойчивости именно этих двух состояний по причине того, что они удовлетворяют условию выравнивания (14.46). Для вектора состояний $(1, -1, 1)$ имеем:

$$\mathbf{Wy} = \frac{1}{3}\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}\begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} +4 \\ -4 \\ +4 \end{bmatrix}.$$

Жестко ограничив этот результат, получим:

$$\operatorname{sgn}[\mathbf{Wy}] = \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} = \mathbf{y}.$$

Аналогично, для вектора состояний $(-1, 1, -1)$ имеем:

$$\mathbf{Wy} = \frac{1}{3}\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}\begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} -4 \\ +4 \\ -4 \end{bmatrix}.$$

Жестко ограничив этот результат, получим:

$$\operatorname{sgn}[\mathbf{Wy}] = \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} = \mathbf{y}.$$

Таким образом, оба вектора состояний удовлетворяют условию выравнивания.

Более того, следуя процедуре асинхронной коррекции (см. табл. 14.2), получим поток, показанный на рис. 14.14, *б*. Эта карта потоков демонстрирует симметрию по отношению к двум устойчивым состояниям сети, что интуитивно понятно. Симметрия является результатом того, что при нулевом индуцированном локальном поле нейрон остается в своем текущем состоянии.

На рис. 14.14, *б* также показано, что если сеть, показанная на рис. 14.14, *а*, находится в начальном состоянии $(1, 1, 1)$, $(-1, -1, 1)$ или $(1, -1, -1)$, то она будет сходиться к устойчивому состоянию $(1, -1, 1)$ за одну итерацию. Если же начальным состоянием будет $(-1, -1, -1)$, $(-1, 1, 1)$ или $(1, 1, -1)$, она будет сходиться к устойчивому состоянию $(-1, 1, -1)$.

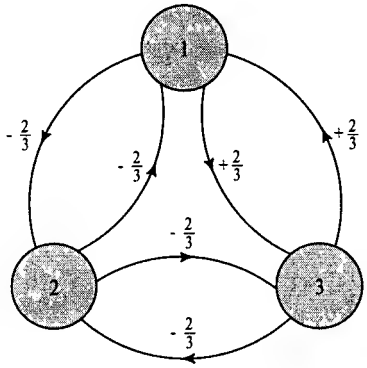
Таким образом, эта сеть содержит две ячейки фундаментальной памяти — $(1, -1, 1)$ и $(-1, 1, -1)$, — представляющие два устойчивых состояния. Применяя выражение (14.43), получим следующую матрицу синаптических весов:

$$\mathbf{W} = \frac{1}{3}\begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix}\begin{bmatrix} +1 & -1 & +1 \end{bmatrix} + \frac{1}{3}\begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix}\begin{bmatrix} -1 & +1 & -1 \end{bmatrix} - \frac{2}{3}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix},$$

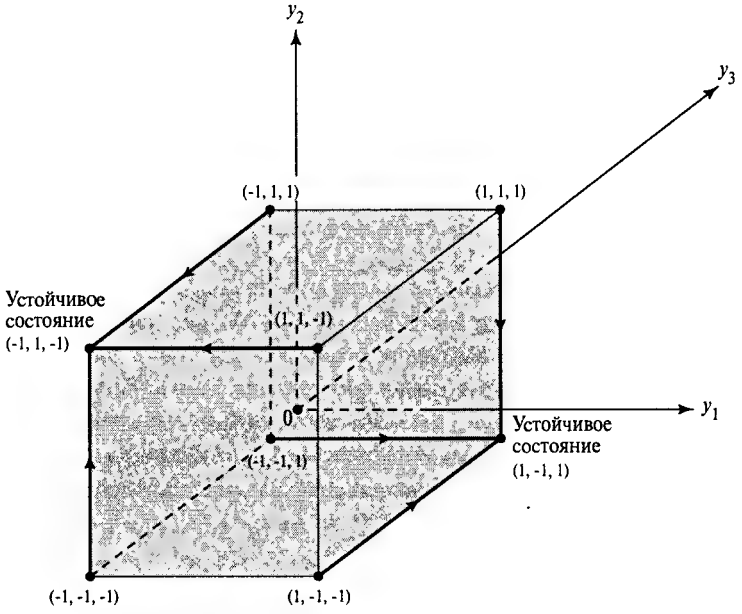
которая проверяется синаптическими весами, показанными на рис. 14.14, *а*.

Способность к коррекции ошибок этой сетью Хопфилда можно ясно увидеть на карте потоков на рис. 14.14, *б*.

1. Если вектор испытания $\boldsymbol{\xi}_{\text{проб.}}$ равен $(1, 1, 1)$, $(-1, -1, 1)$ или $(1, -1, -1)$, результирующим выходом будет фундаментальная память $(1, -1, 1)$. Каждый из этих пробных векторов, по сравнению с сохраненным образом, содержит по одной ошибке.



а)



б)

Рис. 14.14. Архитектурный граф сети Хопфилда, состоящей из $N = 3$ нейронов (а); диаграмма, иллюстрирующая два устойчивых состояния и потоки этой сети (б)

2. Если вектор $\boldsymbol{\xi}_{\text{проб.}}$ равен $(-1,-1,-1)$, $(-1,1,1)$ или $(1,1,-1)$, результирующим выходом будет фундаментальная память $(-1,1,-1)$. Каждый из этих пробных векторов, по сравнению с сохраненным образом, содержит по одной ошибке. ■

Ложные состояния

Матрица весов $\mathbf{W}$ дискретной сети Хопфилда является симметричной (см. выражение (14.44)). Поэтому собственные значения этой матрицы будут действительными

числами. Однако при больших M собственные значения будут вырождаться (degenerate). Это значит, что будут существовать собственные векторы, имеющие одно и то же собственное значение. Собственные векторы, ассоциированные с вырожденными собственными значениями, формируют некоторое подпространство. Более того, если матрица весов $\mathbf{W}$ имеет вырожденное собственное значение, равное нулю, такое подпространство называют нулевым пространством (null space). Нулевое пространство существует исходя из того факта, что количество ячеек фундаментальной памяти M меньше количества нейронов N сети. Наличие нулевого пространства является встроенной характеристикой сети Хопфилда.

Анализ собственных чисел (eigenanalysis) матрицы весов $\mathbf{W}$ приводит к следующей точке зрения на сеть Хопфилда, используемой в качестве ассоциативной памяти [10].

1. Дискретная сеть Хопфилда выступает в качестве векторного проектора (vector projector) — она проектирует пробный вектор в подпространство $\mathbf{M}$, натянутое на векторы фундаментальной памяти.
2. Рассматриваемая динамика сети направляет результирующий вектор проекции к одному из углов единичного гиперкуба, в котором функция энергии минимальна.

Единичный куб имеет N измерений. M векторов фундаментальной памяти, на которые натянуто подпространство $\mathbf{M}$, образуют множество фиксированных точек (устойчивых состояний), находящихся в некоторых углах единичного гиперкуба. Остальные углы единичного гиперкуба, лежащего в подпространстве $\mathbf{M}$, являются потенциальными местоположениями *ложных* (spurious) состояний, также называемых ложными аттракторами (spurious attractor) [42]. Ложные состояния представляют те устойчивые состояния сети Хопфилда, которые отличаются от ячеек фундаментальной памяти этой сети.

В конструкции сети Хопфилда, выступающей в качестве ассоциативной памяти, приходится сталкиваться с необходимостью удовлетворения двух противоречивых требований: необходимостью сохранения векторов фундаментальной памяти как фиксированных точек пространства состояний и желанием иметь как можно меньше ложных состояний.

Емкость сети Хопфилда

К сожалению, ячейки фундаментальной памяти сети Хопфилда не всегда являются устойчивыми. Более того, при этом могут появляться ложные состояния, являющиеся устойчивыми состояниями, отличными от ячеек фундаментальной памяти. Эти два явления понижают эффективность сети Хопфилда, выступающей в качестве ассоциативной памяти. В этом подразделе остановимся на первом из этих явлений.

Применим пробный вектор, равный одной из ячеек фундаментальной памяти $\boldsymbol{\xi}_v$, к нашей сети. Тогда, разрешив использование собственных обратных связей и предполагая нулевое внешнее смещение, с помощью выражения (14.41) можно определить индуцированное локальное поле нейрона j следующим образом:

$$v_j = \sum_{i=1}^{N} w_{ji}\xi_{v,i} = \frac{1}{N}\sum_{\mu=1}^{N}\xi_{\mu,j}\sum_{i=1}^{N}\xi_{\mu,i}\xi_{v,i} = \xi_{v,j} + \frac{1}{N}\sum_{\mu=1,\mu\neq v}^{N}\xi_{\mu,j}\sum_{i=1}^{N}\xi_{\mu,i}\xi_{v,i}. \quad (14.47)$$

Первое слагаемое в правой части (14.47) является j-м элементом ячейки фундаментальной памяти $\boldsymbol{\xi}_v$; теперь ясно, для чего в определении синаптического веса w_{ji} (14.41) был введен масштабирующий множитель $1/N$. Таким образом, это слагаемое можно рассматривать как желаемый "сигнал" компонента v_j. Второе слагаемое в правой части (14.47) является результатом "переговоров" между элементами фундаментальной памяти $\boldsymbol{\xi}_v$, подвергающихся тестированию, и элементами других ячеек фундаментальной памяти $\boldsymbol{\xi}_\mu$. Это значит, что второе слагаемое можно рассматривать как "шумовой" компонент индуцированного локального поля v_j. Следовательно, приходим к ситуации, аналогичной классической задаче "извлечения сигнала из шума" из теории связи [437].

Предположим, что ячейки фундаментальной памяти являются случайными и генерируются как последовательность MN *попыток Бернулли* (Bernoulli trial). Тогда слагаемое шума в выражении (14.47) состоит из суммы $N(M-1)$ независимых случайных переменных, принимающих значения $\pm 1/N$. Это — ситуация, в которой применима центральная предельная теорема теории вероятностей. Эта теорема гласит следующее [294].

Пусть $\{X_k\}$ — последовательность взаимно независимых случайных переменных, имеющих общее распределение. Предположим, что X_k имеет среднее значение μ и дисперсию σ^2 и что $Y = X_1 + X_2 + \ldots + X_n$. Тогда при устремлении n к бесконечности случайная переменная суммы Y достигнет гауссова распределения.

Тогда, применяя центральную предельную теорему к слагаемому шума в (14.47), получим, что шум является асимптотически распределенным по Гауссу. Каждая из $N(M-1)$ случайных переменных, составляющих слагаемое шума в этом равенстве, имеет нулевое среднее значение и дисперсию, равную $1/N^2$. Отсюда следует, что статистика полученного гауссова распределения будет следующей.

- Среднее значение будет равно нулю.
- Дисперсия будет равна $(M-1)/N$.

Компонент сигнала $\xi_{v,j}$ с равной вероятностью может принимать значения $+1$ и -1 и, следовательно, имеет нулевое среднее и единичную дисперсию. Исходя из этого, отношение сигнал/шум можно определить следующим образом:

$$\rho = \frac{\text{дисперсия сигнала}}{\text{дисперсия шума}} = \frac{1}{(M-1)/N} \simeq \frac{N}{M} \quad \text{для больших } M. \tag{14.48}$$

Компоненты фундаментальной памяти ξ_v будут устойчивыми тогда и только тогда, когда отношение сигнал/шум будет высоким. Количество M ячеек фундаментальной памяти является прямой мерой емкости памяти (storage capacity) сети. Таким образом, из (14.48) следует, что, до тех пор, пока емкость памяти сети не будет превышена (т.е. количество M ячеек фундаментальной памяти будет оставаться малым по сравнению с количеством N нейронов сети), фундаментальная память будет устойчивой в вероятностном смысле.

Величина, обратная к отношению сигнал/шум, т.е.

$$\alpha = \frac{M}{N}, \tag{14.49}$$

называется параметром загрузки (load parameter). Соглашения статистической физики утверждают, что качество извлечения из памяти сети Хопфилда ухудшается при увеличении параметра загрузки α и преломляется (breaks down) при критичном значении $\alpha_c = 0,14$ [42], [760]. Эта оценка критичного значения была получена в [480] в результате компьютерного моделирования, в котором одновременно могло извлекаться $0,15N$ состояний до появления ошибок.

При $\alpha_c = 0,14$ из выражения (14.48) можно получить, что критичное значение отношения сигнал/шум приблизительно равно $\rho_c \approx 7$, или 8,45 dB. Если соотношение сигнал/шум находится ниже этой критической отметки, восстановление данных из памяти становится невозможным.

Критическое значение

$$M_c = \alpha_c N = 0,14N \tag{14.50}$$

определяет емкость памяти с ошибками (storage capacity with errors) при вспоминании. Для определения емкости памяти без ошибок следует использовать более строгий критерий, определяемый в терминах вероятности ошибок, который будет описан ниже.

Пусть j-й бит испытания $\boldsymbol{\xi}_{\text{проб.}} = \boldsymbol{\xi}_v$ будет символом 1 (т.е. $\xi_{v,j} = 1$). Тогда условная вероятность ошибки в бите при вспоминании (conditional probability of bit error on recall) будет определяться областью, которая закрашена на рис. 14.15. Оставшаяся часть области под кривой является условной вероятностью того, что бит j этого ис-

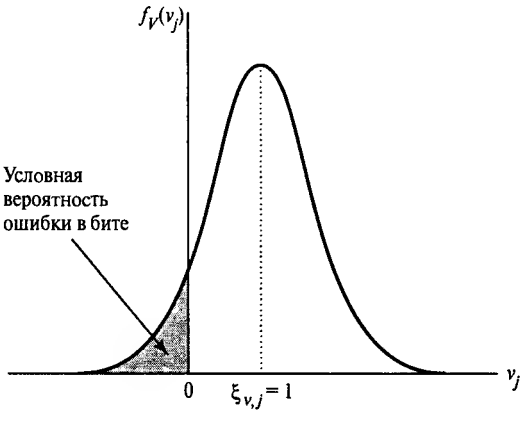


Рис. 14.15. Условная вероятность ошибки в бите в предположении гауссова распределения индуцированного локального поля v_j нейрона j. (Нижний индекс V в обозначении функции плотности вероятности $f_V(v_j)$ обозначает случайную переменную, реализацией которой является v_j.)

пытания будет вспомнен корректно. Используя хорошо известную формулу гауссова распределения, последняя условная вероятность будет равна

$$P(v_j > 0|\xi_{v,j} = +1) = \frac{1}{\sqrt{2\pi}\sigma}\int_0^\infty \exp\left(-\frac{(v_j-\mu)^2}{2\sigma^2}\right)dv_j. \tag{14.51}$$

Если $\xi_{v,j} = +1$ и среднее значение слагаемого шума в (14.47) равно нулю, то среднее μ случайной переменной V будет равно единице, а ее дисперсия $\sigma^2 = (M-1)/N$. Из определения функции ошибки, как правило, использующейся в вычислениях, связанных с гауссовыми распределениями, имеем:

$$\mathrm{erf}(y) = \frac{2}{\sqrt{\pi}}\int_0^y e^{-z^2}dz, \tag{14.52}$$

где y — переменная, определяющая верхний предел интегрирования. Теперь можно упростить выражения для условной вероятности корректного вспоминания j-го бита фундаментальной памяти $\boldsymbol{\xi}_v$, переписав (14.51) в терминах функции ошибки:

$$P(v_j > 0|\xi_{v,j} = +1) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\sqrt{\frac{\rho}{2}}\right)\right], \tag{14.53}$$

где ρ — отношение сигнал/шум, определяемое выражением (14.48). Каждая ячейка фундаментальной памяти состоит из n бит. Ячейки фундаментальной памяти также всегда равновероятны. Отсюда следует, что вероятность устойчивости образов определяется соотношением

$$p_{\text{стаб.}} = (P(v_j > 0|\xi_{v,j} = +1))^N. \tag{14.54}$$

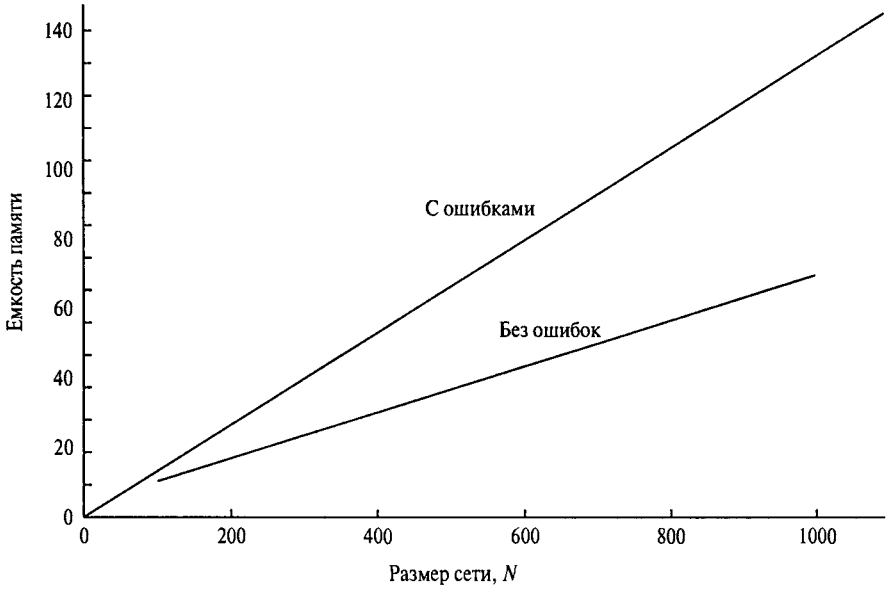


Рис. 14.16. Графики емкости памяти сети Хопфилда по отношению к ее размеру для двух случаев: вспоминания с ошибками и вспоминания практически без ошибок

Эту вероятность можно использовать для вывода выражения емкости сети Хопфилда. В частности, емкость памяти с вспоминанием практически без ошибок (storage capacity almost without errors) определяем как наибольшее количество ячеек фундаментальной памяти, которые могут сохраняться в сети при требовании, чтобы подавляющее большинство из них вспоминалось корректно. В задаче 14.8 показано, что из такого определения вытекает следующая формула:

$$M_{\max} = \frac{N}{2\log_e N}, \tag{14.55}$$

где $\log_e$ — натуральный логарифм.

На рис. 14.16 показаны графики емкости памяти без ошибок, определяемой соотношением (14.50), и емкости памяти, практически не содержащей ошибок, определяемой выражением (14.55). Оба графика построены относительно размера сети N. На этом рисунке можно заметить следующее.

- Емкость памяти сети Хопфилда растет *линейно* по отношению к размеру сети N.
- Главное ограничение сети Хопфилда состоит в том, что для восстанавливаемости фундаментальной памяти ее емкость памяти должна поддерживаться на относительно низком уровне[6].

14.8. Компьютерное моделирование 1

В этом разделе проведем компьютерное моделирование для иллюстрации поведения дискретной сети Хопфилда, выступающей в качестве ассоциативной памяти. В эксперименте будет использоваться сеть, состоящая из $N = 120$ нейронов и, следовательно, имеющая $N^2 - N = 12280$ синаптических весов. Эта сеть обучалась для извлечения восьми цифроподобных черно-белых образов, приведенных на рис. 14.7. Каждый из этих рисунков содержит 120 пикселей (элементов). Эти образы создавались специально для достижения сетью хорошей производительности [657]. Для входов, применяемых к сети, предполагалось, что белому цвету соответствует значение $+1$, а черному — -1. Каждый из образов на рис. 14.17 использовался как ячейка фундаментальной памяти на фазе запоминания (обучения) сети Хопфилда для создания матрицы синаптических весов $\mathbf{W}$ с помощью формулы (14.43). Фаза вспоминания осуществлялась в асинхронном режиме, согласно процедуре, описанной в табл. 14.2.

---

[6] **Немонотонная функция активации**

Для преодоления ограничений модели Хопфилда, работающей в качестве ассоциативной памяти, в литературе предлагалось множество подходов. Пожалуй, самым существенным из них на данный момент является улучшение непрерывной модели Хопфилда, приведенное в [755]. Эта модификация ограничивает функцию активации $\varphi(\cdot)$ нейрона, остааляя, таким образом, нетронутой простоту конструкции сети. В частности, ограниченные или сигмоидальные функции активации всех нейронов сети заменяются немонотонной функцией. В математических терминах эта функция определяется как следующее произведение:

$$\varphi(v) = \left(\frac{1 - \exp(-av)}{1 + \exp(-av)}\right)\left(\frac{1 + \kappa\exp(b(|v| - c))}{1 + \exp(b(|v| - b))}\right), \tag{1}$$

где v — индуцированное локальное поле. Первый множитель в правой части (1) является обычной сигмоидальной функцией (гиперболическим тангенсом), используемой в обычных сетях Хопфилда; параметр a является коэффициентом передачи нейрона. Второй множитель отвечает за придание функции активации немонотонного вида. Оба параметра, характеризующие этот второй множитель (b и c), являются положительными константами; оставшийся параметр κ — обычно отрицательный. В экспериментах, описанных в [755], использовались следующие значения этих параметров:

$$a = 50;\ b = 15,$$
$$c = 0,5;\ \kappa = -1.$$

Согласно вышеназванной работе, точная форма функции активации и параметров, используемых для описания модели, не столь критична. Самым существенным здесь является сам факт немонотонности функции активации.

Эта модель ассоциативной памяти обладает двумя интересными свойствами [1174].

В сети, составленной из N нейронов, емкость памяти модели составляет более $0,3N$, что при больших N значительно больше емкости обычной модели Хопфилда ($N/2\ log\ N$).

В этой модели нет ложных состояний. Если не удается восстановить какую-либо корректную ячейку памяти, состояние сети переходит к хаотическому поведению. (Понятие хаоса рассматривалось в разделе 14.13.)

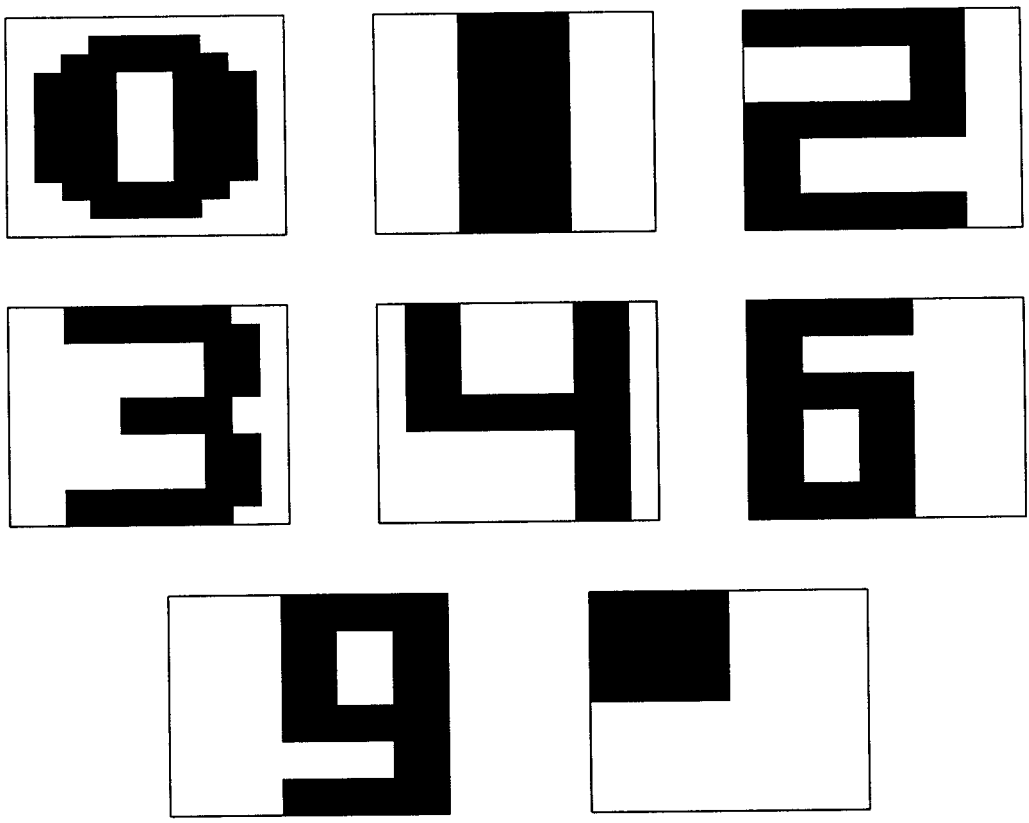


Рис. 14.17. Множество рукописных образов, используемых при компьютерном моделировании сети Хопфилда

Во время первой части фазы вспоминания на вход сети подавались ячейки фундаментальной памяти. Таким образом, выполнялась проверка способности их корректного восстановления из информации, сохраненной в матрице синаптических весов. В каждом из случаев желаемый образ восстанавливался сетью после одной итерации.

Далее, для того чтобы продемонстрировать способность сетей Хопфилда корректировать ошибки, рассматриваемый образ преднамеренно искажался при помощи обращения случайно и независимо выбираемых пикселей (значения этих пикселей изменялись с $+1$ на -1 и с -1 на $+1$ с вероятностью 0,25), после чего подавался на вход сети. Результат этого эксперимента для цифры 3 показан на рис. 14.18. Образ, показанный на этом рисунке в центре верхнего ряда, представляет собой искаженную версию цифры 3. Этот образ в момент времени 0 подавался на вход сети. Образы, произведенные сетью после 5, 10, 15, 20, 25, 30 и 35 итераций, показаны в остальных фрагментах этого рисунка. По мере увеличения количества итераций заметно постепенное увеличение сходства между выходом сети и цифрой 3. Видно, что после 35 итераций сеть сошлась к абсолютно корректной форме цифры 3.

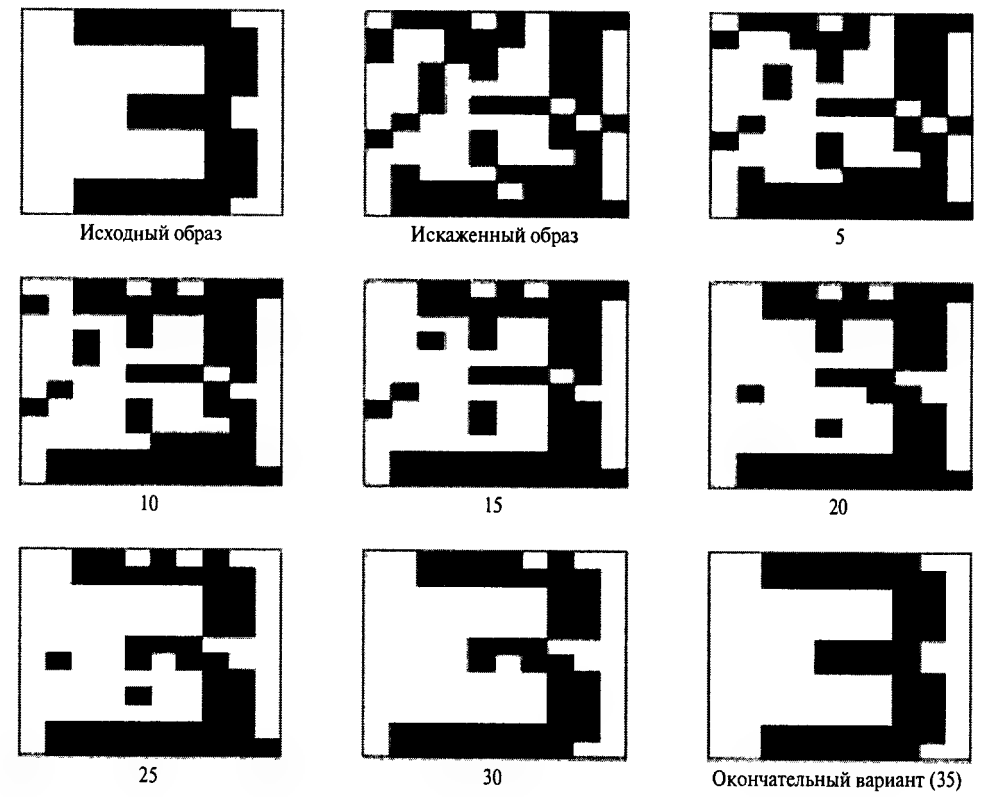


Рис. 14.18. Корректировка искаженного образа цифры 3

При условии, что теоретически одна четверть из 120-ти нейронов сети Хопфилда меняла свое состояние в каждом искаженном образе, среднее значение количества итераций, необходимых для вспоминания, составляет 30. В нашем эксперименте количество итераций, необходимых для восстановления различных образов из их искаженных версий, приведено ниже.

Образ	Количество итераций, потребовавшихся для восстановления
0	34
1	32
2	26
3	35
4	25
6	37
точка	32
9	26

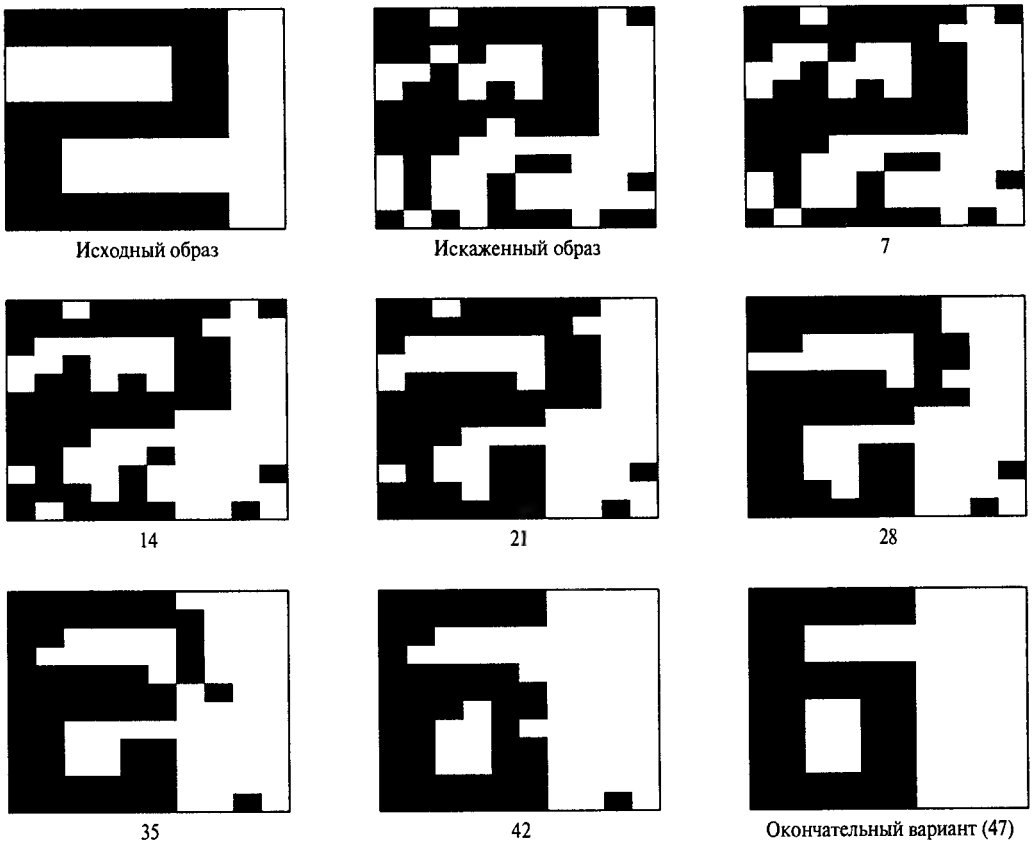


Рис. 14.19. Некорректное восстановление образа цифры 2

Таким образом, среднее количество итераций, необходимых для восстановления, немного превысило 31. Это показывает, что сеть Хопфилда вела себя так, как ожидалось.

Внутренняя проблема сетей Хопфилда возникает в том случае, когда сети представляется искаженная версия некоторой ячейки фундаментальной памяти, а сеть продолжает сходиться к некорректной ячейке (см. рис. 14.19). В этом случае сети подавался на вход искаженный образ цифры 2, а после 47 итераций он сошелся к ячейке фундаментальной памяти цифры 6.

Как уже говорилось, существует еще одна проблема, возникающая в сетях Хопфилда, — это наличие ложных состояний. На рис. 14.20 (рассматриваемом как матрица, состоящая из 14 × 8 состояний сети) представлен список 108 ложных аттракторов, найденных после 43097 тестов случайно выбираемых цифр, имеющих искажения отдельных бит с вероятностью 0,25. Ложные состояния можно сгруппировать следующим образом [42].

1. Обратные ячейки фундаментальной памяти. Эти ложные состояния представляют собой обратные (т.е. взятые с противоположным знаком) версии ячеек фундаментальной памяти сети (см. первую ячейку в первом ряду рис. 14.20, которая обратна образу цифры 6 на рис. 14.17). Для описания этого типа ложных состояний заметим, что функция энергии E является симметричной, т.е. ее значения не изменяются, если состояния всех нейронов обращаются (т.е. для всех i состояние x_i заменяется на $-x_i$). Следовательно, если ячейка фундаментальной памяти $\boldsymbol{\xi}_\mu$ соответствует определенному локальному минимуму поверхности энергии, то этому же локальному минимуму соответствует и ячейка $-\boldsymbol{\xi}_\mu$. Обратный знак не составляет проблемы при извлечении информации, если принять соглашение обращать все информационные биты восстановленного образца в случае, когда оказывается, что вместо "знакового" бита "+1" стоит бит "−1".
2. Смешанные состояния (mixture state). Смешанные ложные состояния представляют собой линейные комбинации нечетного количества сохраненных образов. Например, рассмотрим состояние

$$x_i = \operatorname{sgn}(\xi_{1,i} + \xi_{2,i} + \xi_{3,i}),$$

являющееся смесью трех состояний. Это состояние сформировано из трех ячеек фундаментальной памяти $\boldsymbol{\xi}_1$, $\boldsymbol{\xi}_2$ и $\boldsymbol{\xi}_3$ по правилу большинства (majority rule). Условие устойчивости (14.45) в больших сетях удовлетворяется таким состоянием. Состояние в 6 ряду и 4 столбце рис. 14.20 является комбинацией трех состояний, сформированной следующими состояниями: $\boldsymbol{\xi}_1$ = (образ, обратный единице), $\boldsymbol{\xi}_2$ = цифра 4 и $\boldsymbol{\xi}_3$ = цифра 9.
3. Зеркальные состояния (spin-glass state). Этот тип ложных состояний получил свое название по аналогии с зеркальными моделями в статистической механике. Зеркальные состояния определяются локальными минимумами поверхности (landscape) энергии, которые не коррелированы с какими-либо ячейками фундаментальной памяти сети (см. состояние в 7 ряду и 6 столбце на рис. 14.20).

14.9. Теорема Коэна–Гроссберга

В [202] рассматривается общий принцип достижения устойчивости определенным классом нейронных сетей, описываемых следующей системой нелинейных дифференциальных уравнений:

$$\frac{d}{dt}u_j = a_j(u_j)\left[b_j(u_j) - \sum_{i=1}^{N} c_{ji}\varphi_i(u_i)\right], \; j = 1, \ldots, N. \tag{14.56}$$

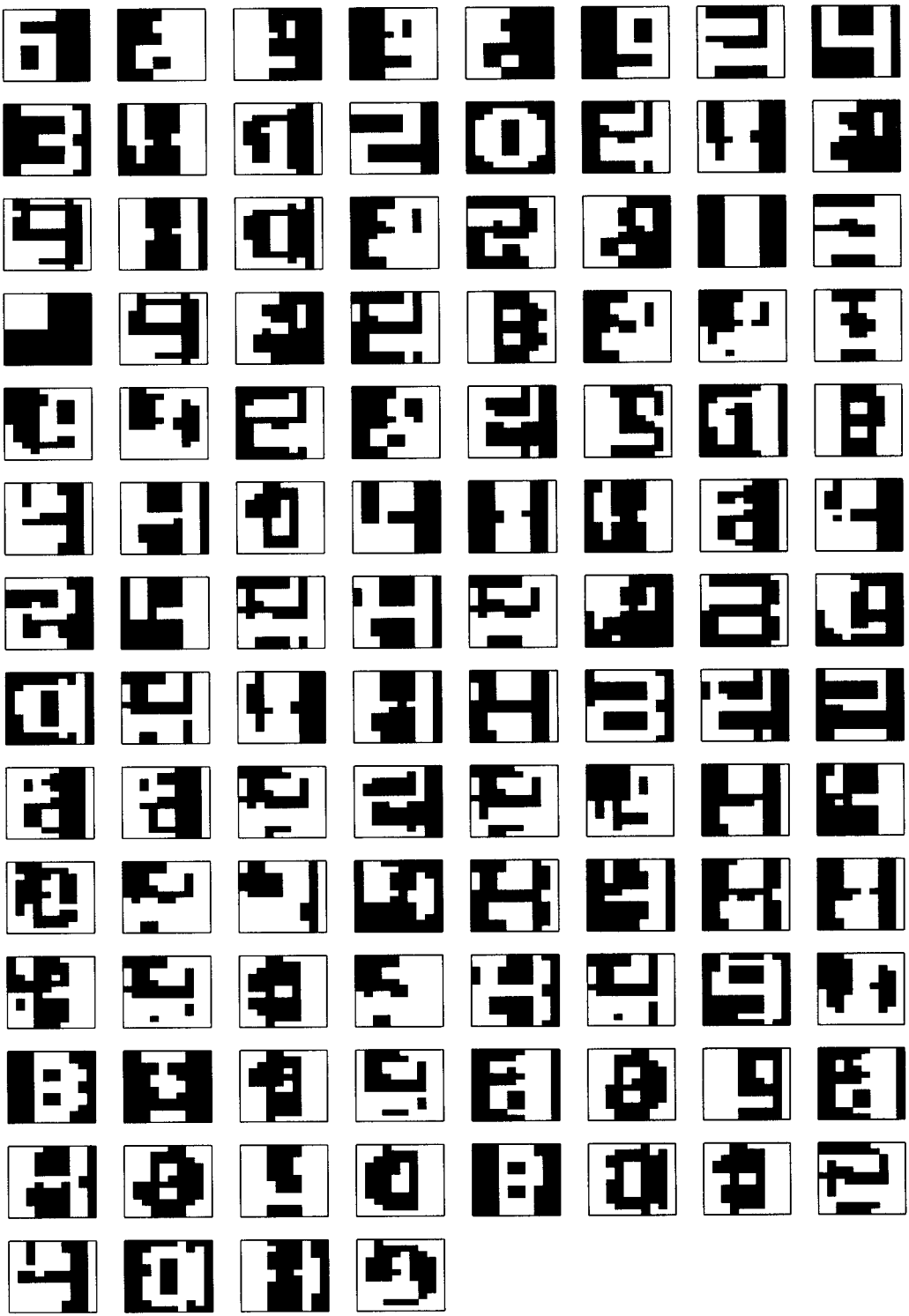


Рис. 14.20. Перечень ложных состояний для сети Хопфилда в компьютерном эксперименте 1

Согласно этой работе, данный класс нейронных сетей допускает определение функции Ляпунова следующим образом (см. задачу 14.13):

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ji}\varphi_i(u_i)\varphi_j(u_j) - \sum_{j=1}^{N}\int_0^{u_j} b_j(\lambda)\varphi_j'(\lambda)d\lambda, \tag{14.57}$$

где

$$\varphi_j'(\lambda) = \frac{d}{d\lambda}(\varphi_j(\lambda)). \tag{14.58}$$

Однако, для того чтобы определение (14.57) было корректным, необходимо выполнение следующих условий.

1. Условие симметрии. Синаптические веса сети должны быть симметричными:

$$c_{ij} = c_{ji}. \tag{14.59}$$

2. Условие неотрицательности. Функция $a_j(u_j)$ должна быть неотрицательной:

$$a_j(u_j) \geq 0. \tag{14.60}$$

3. Условие монотонности. Нелинейная функция отображения входа на выход $\varphi_j(u_j)$ должна быть монотонной:

$$\varphi_j'(u_j) = \frac{d}{du_j}\varphi_j(u_j) \geq 0. \tag{14.61}$$

Теперь можно сформулировать теорему Коэна–Гроссберга (Cohen-Grossberg theorem).

Если система нелинейных дифференциальных уравнений (14.56) удовлетворяет условиям симметрии, неотрицательности и монотонности, то функция Ляпунова системы (14.57) удовлетворяет условию

$$\frac{dE}{dt} \leq 0.$$

Если функция Ляпунова E обладает этим глобальным свойством, то, согласно первой теореме Ляпунова, эта система является глобально устойчивой.

ТАБЛИЦА 14.3. Соответствие между теоремой Коэна–Гроссберга и моделью Хопфилда

Теорема Коэна–Гроссберга	Модель Хопфилда
u_j	$C_j v_j$
$a_j(u_j)$	1
$b_j(u_j)$	$-(v_j/R_j) + I_j$
c_{ji}	$-w_{ji}$
$\varphi_i(u_i)$	$\varphi_i(v_i)$

Модель Хопфилда как частный случай теоремы Коэна–Гроссберга

Сравнивая общую систему уравнений (14.56) с системой (14.20) для непрерывной модели Хопфилда, можно установить соответствие между этими моделями (табл. 14.3). Подставляя значения из табл. 14.3 в уравнение (14.57), получим следующую функцию Ляпунова для непрерывной модели Хопфилда:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ji}\varphi_i(v_i)\varphi_j(v_j) + \sum_{j=1}^{N}\int_0^{v_j}\left(\frac{v_j}{R_j} - I_j\right)\varphi_j'(v)dv, \tag{14.62}$$

где нелинейная функция активации $\varphi(\cdot)$ определяется выражением (14.23).

Затем можно сделать следующие наблюдения.

1. $$\varphi_i(v_i) = x_i.$$

2. $$\int_0^{v_j}\varphi_j'(v)dv = \int_0^{x_j} dx = x_j.$$

3. $$\int_0^{v_j} v\varphi_j'(v)dv = \int_0^{x_j} dx = \int_0^{x_j}\varphi_j^{-1}(x)dx.$$

Выражения 2 и 3 вытекают из подстановки $x = \varphi_i(v)$. Таким образом, подставляя эти выражения в функцию Ляпунова (14.62), получим результат, идентичный полученному в выражении (14.28). Однако обратим внимание на то, что для существования обобщенной функции Ляпунова (14.62) $\varphi_i(v)$ должна быть неубывающей функцией входа v, но не обязательно должна иметь обратную.

Теорема Коэна–Гроссберга является общим принципом нейродинамики для широкого спектра областей применения [387]. В следующем разделе рассмотрим еще одно применение этой важной теоремы.

14.10. Модель BSB

В этом разделе мы продолжим нейродинамический анализ ассоциативной памяти и изучим модель состояния мозга BSB (brain-state-in-box), которая впервые была описана в [54]. В своей основе модель BSB является системой с положительной обратной связью с амплитудным ограничением (positive feedback system with amplitude limitation). Она состоит из множества взаимосвязанных нейронов, которые замкнуты обратной связью сами на себя. В [54] эта модель использует встроенные обратные связи для *усиления* (amplify) входного образа до тех пор, пока все нейроны сети не достигнут насыщения. Таким образом, модель BSB можно рассматривать как устройство классификации, в котором аналоговому входу дается цифровое представление, определяемое некоторым устойчивым состоянием модели.

Пусть $\mathbf{W}$ — симметричная матрица весов, наибольшие собственные значения которой имеют положительные действительные части. Пусть $\mathbf{x}(0)$ — вектор начальных состояний этой модели, представляющий некоторый входной сигнал активации. Предполагая, что модель имеет N нейронов, вектор состояния этой модели имеет размерность N, а матрица весов $\mathbf{W}$ — размерность $N \times N$. Алгоритм BSB полностью определяется следующей парой уравнений:

$$\mathbf{y}(n) = \mathbf{x}(n) + \beta \mathbf{W}\mathbf{x}(n), \tag{14.63}$$

$$\mathbf{x}(n+1) = \varphi(\mathbf{y}(n)), \tag{14.64}$$

где β — малая положительная константа, называемая коэффициентом обратной связи (feedback factor); $\mathbf{x}(n)$ — вектор состояний модели в дискретный момент времени n. На рис. 14.21, *а* показана блочная диаграмма системы уравнений (14.63) и (14.64); блок с меткой $\mathbf{W}$ представляет однослойную линейную нейронную сеть (см. рис. 14.21, *б*). Функция активации φ является кусочно-линейной функцией аргумента $y_j(n)$ — j-го компонента вектора $\mathbf{y}(n)$ (рис. 14.22):

$$x_j(n+1) = \varphi(y_j(n)) = \begin{cases} +1, & \text{если } y_j(n) > +1, \\ y_j(n), & \text{если } -1 \le y_j(n) \le +1, \\ -1, & \text{если } y_j(n) < -1. \end{cases} \tag{14.65}$$

Равенство (14.65) ограничивает вектор состояний модели BSB N-мерным единичным кубом с центром в начале координат.

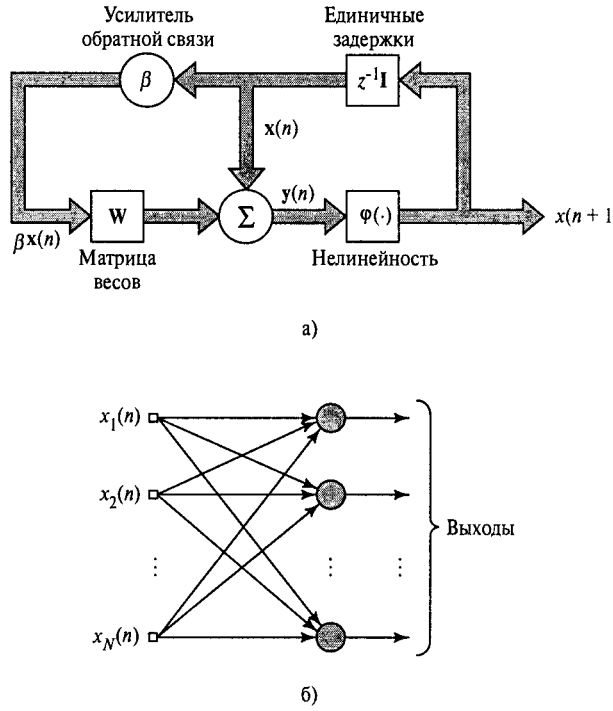


Рис. 14.21. Блочная диаграмма модели BSB (а); граф передачи сигнала линейного ассоциатора с матрицей весов **W** (б)

Итак, алгоритм работает следующим образом. В качестве вектора начального состояния на вход модели BSB подается сигнал активации $\mathbf{x}(0)$, после чего для вычисления вектора $\mathbf{y}(0)$ используется уравнение (14.63). Затем для усечения вектора $\mathbf{y}(0)$ используется равенство (14.64), в результате чего получается вектор $\mathbf{x}(1)$. Далее, применяя в цикле выражения (14.63) и (14.64), получим $\mathbf{x}(2)$. Этот цикл продолжается до тех пор, пока модель BSB не достигнет некоторого устойчивого состояния, представленного одной из вершин единичного гиперкуба. Интуитивно понятно, что положительная обратная связь в модели BSB приводит к увеличению Евклидовой длины (нормы) вектора начального состояния $\mathbf{x}(0)$ до тех пор, пока вектор состояния не упрется в стенку единичного гиперкуба. После этого вектор продолжает "скольжение" по стенке куба, пока не достигнет некоторого его "устойчивого" угла, где он пытается выйти наружу, чему препятствует единичный куб [546]. Описанный процесс положен в основу названия данной модели, которое дословно переводится как "модель мозга в виде ящика".

Функция Ляпунова модели BSB

Модель BSB можно переопределить как частный случай нейродинамической модели, описанной выражением (14.16) [387]. Для того чтобы это увидеть, вначале перепишем

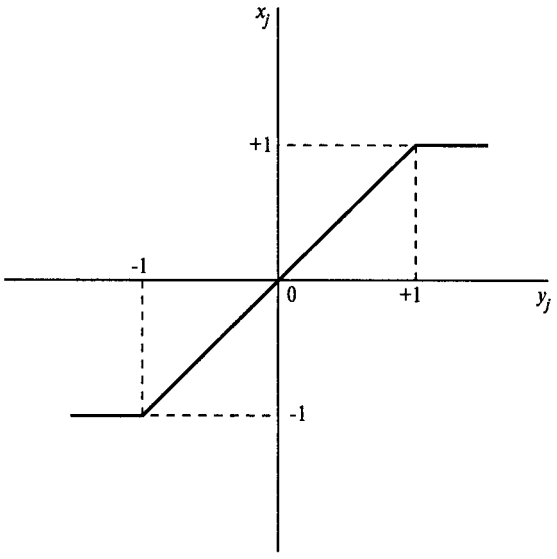


Рис. 14.22. Кусочно-линейная активационная функция, используемая в модели BSB

j-й компонент алгоритма BSB (14.63) и (14.64) в следующем виде:

$$x_j(n+1) = \varphi\left(\sum_{i=1}^{N} c_{ji}x_i(n)\right), \quad j = 1, 2, \ldots, N. \tag{14.66}$$

Коэффициенты c_{ji} определяются таким образом:

$$c_{ji} = \delta_{ji} + \beta w_{ji}, (14.67) \tag{14.67}$$

где δ_{ji} — дельта Кронекера, равная единице при $i = j$ и нулю в противном случае; w_{ji} — ji-й элемент матрицы весов $\mathbf{W}$. Уравнение (14.66) записано в форме для дискретного времени. Чтобы продолжить, потребуется переформулировать его в форме непрерывного времени:

$$\frac{d}{dt}x_j(t) = -x_j(t) + \varphi\left(\sum_{i=1}^{N} c_{ji}x_i(t)\right), \quad j = 1, 2, \ldots, N, \tag{14.68}$$

где смещение I_j равно нулю для всех j. Однако, для того чтобы применить теорему Коэна–Гроссберга, требуется сделать еще один шаг и преобразовать (14.68) в форму аддитивной модели. Это можно сделать, введя новое множество переменных:

$$v_j(t) = \sum_{i=1}^{N} c_{ji}x_i(t). \tag{14.69}$$

ТАБЛИЦА 14.4. Соответствие между теоремой Коэна–Гроссберга и моделью BSB

Теорема Коэна–Гроссберга	*Модель BSB*
u_j	v_j
$a_j(u_j)$	1
$b_j(u_j)$	$-v_j$
c_{ji}	$-c_{ji}$
$\varphi_j(u_j)$	$\varphi_j(v_j)$

Тогда с помощью определения (14.67) получим:

$$x_j(t) = \sum_{i=1}^{N} c_{ji} v_i(t). \tag{14.70}$$

Следовательно, модель (14.68) можно переписать в эквивалентной форме:

$$\frac{d}{dt} v_j(t) = -v_j(t) + \sum_{i=1}^{N} c_{ji} \varphi(v_i(t)), \quad j = 1, 2, \ldots, N. \tag{14.71}$$

Теперь можно применить теорему Коэна–Гроссберга к модели BSB. Сравнивая (14.71) и (14.56), мы видим соответствия между моделью BSB и теоремой Коэна (табл. 14.4).

Таким образом, подставив результаты из табл. 14.4 в выражение (14.57), получим, что функция Ляпунова для модели BSB имеет следующий вид:

$$E = -\frac{1}{2} \sum_{j=1}^{N} \sum_{i=1}^{N} c_{ji} \varphi(v_j) \varphi(v_i) + \sum_{j=1}^{N} \int_0^{v_j} v \varphi'(v) dv, \tag{14.72}$$

где $\varphi'(v)$ — первая производная сигмоидальной функции $\varphi(v)$ по своему аргументу. В заключение, подставляя определения (14.65), (14.67) и (14.69) в (14.72), можно определить функцию Ляпунова (энергии) для модели BSB в терминах исходных переменных состояний [387]:

$$E = -\frac{\beta}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ji} x_j x_i = -\frac{\beta}{2} \mathbf{x}^T \mathbf{W} \mathbf{x}. \tag{14.73}$$

Оценка функции Ляпунова для сети Хопфилда, представленная в разделе 14.7, предполагает существование производной функции, обратной к сигмоидальной нелинейности модели. Это требование удовлетворяется для функции гиперболического тангенса. Однако этому условию не удовлетворяет модель BSB, если переменная состояния j-го нейрона в ней может принимать только значения +1 и –1. Несмотря на

эту сложность, функцию Ляпунова для модели BSB можно вычислить с помощью теоремы Коэна–Гроссберга. Этот факт хорошо иллюстрирует общую применимость этой важной теоремы.

Динамика модели BSB

При непосредственном анализе, выполненном в [366], было показано, что модель BSB является де-факто алгоритмом наискорейшего спуска, который минимизирует энергию E, определяемую формулой (14.73). Это важное свойство модели BSB, однако, предполагает, что матрица весов $\mathbf{W}$ удовлетворяет следующим условиям.

- Матрица $\mathbf{W}$ симметрична:

$$\mathbf{W} = \mathbf{W}^T.$$

- Матрица $\mathbf{W}$ является положительно полуопределенной, т.е. в терминах собственных значений:

$$\lambda_{\min} \geq 0,$$

где $\lambda_{\min}$ — наименьшее собственное значение матрицы $\mathbf{W}$.

Исходя из этого, функция энергии E модели BSB убывает с ростом n (количество итераций), если вектор состояний $\mathbf{x}(n+1)$ в момент времени $n+1$ отличается от вектора $\mathbf{x}(n)$ в момент времени n. Более того, точки минимума функции энергии E определяют равновесные состояния модели BSB, которые характеризуются следующим:

$$\mathbf{x}(n+1) = \mathbf{x}(n).$$

Другими словами, подобно модели Хопфилда, модель BSB является сетью, минимизирующей энергию (energy-minimizing network).

Равновесные состояния модели BSB определены в некоторых углах единичного гиперкуба и в начале координат. В последнем случае любые отклонения вектора состояний, независимо от того, насколько малыми они будут, усиливаются положительными обратными связями модели и, таким образом, уводят состояние модели от начала координат в направлении некоторой устойчивой конфигурации. Другими словами, начало координат является седловой точкой. Для того чтобы все углы единичного гиперкуба являлись возможными равновесными состояниями модели BSB, необходимо выполнение третьего условия [382].

- Матрица весов **W** должна быть диагонально-доминантной (diagonal dominant), т.е.

$$w_{jj} \geq \sum_{i \neq j} |w_{ij}|, \quad j = 1, 2, \ldots, N, \tag{14.74}$$

где w_{ij} — ij-й элемент матрицы **W**.

Для того чтобы равновесное состояние **x** было устойчивым (т.е. чтобы угол единичного гиперкуба был фиксированной точкой аттрактора), в этом кубе должен существовать бассейн аттракции **N(x)**, такой, чтобы для любого вектора начального состояния **x**(0) из **N(x)** модель BSB сходилась к **x**. Для того чтобы все углы единичного гиперкуба были возможными точечными аттракторами, матрица весов **W** должна удовлетворять четвертому условию [382].

- Матрица весов **W** должна быть строго диагонально-доминантной, т.е.

$$w_{jj} \geq \sum_{i \neq j} |w_{ij}| + \alpha, \quad j = 1, 2, \ldots, N, \tag{14.75}$$

где α — некоторая положительная константа.

Важной точкой в этой дискуссии является то, что если модель BSB имеет симметричную и положительно полуопределенную матрицу весов **W** (что чаще всего и происходит), то только некоторые (но не все) углы единичного гиперкуба являются точечными аттракторами. Для того чтобы все углы были потенциальными точечными аттракторами, матрица весов **W** должна также удовлетворять условию (14.75), которое включает в себя условие (14.74).

Кластеризация

Естественным применением модели BSB является кластеризация (clustering). Это следует из того факта, что устойчивые углы единичного гиперкуба выступают в роли точечных аттракторов с четко определенными бассейнами аттракции. Эти бассейны делят пространство состояний на соответствующее множество четко очерченных областей. Следовательно, модель BSB может использоваться в качестве алгоритма кластеризации без учителя (unsupervised clustering algorithm), в котором все устойчивые углы единичного гиперкуба представляют собой "кластеры" рассматриваемых данных. Самоусиление, производимое положительными обратными связями (соответствующее первому принципу самоорганизации, сформулированному в главе 8), является важной составляющей этого свойства кластеризации.

В [56] описано использование модели BSB для кластеризации (и, следовательно, идентификации) сигналов радара от разных источников. В этом приложении матрица весов $\mathbf{W}$, являющаяся основой работы модели BSB, обучалась с использованием линейного ассоциатора (ассоциативной памяти), обучаемого методом коррекции ошибок (см. главу 2). Для примера предположим, что информация представлена множеством K векторов обучения, которые ассоциированы сами с собой следующим образом:

$$\mathbf{x}_k \rightarrow \mathbf{x}_k, k = 1, 2, \ldots, K. \tag{14.76}$$

Пусть вектор обучения $\mathbf{x}_k$ выбирается случайным образом. Тогда матрица весов $\mathbf{W}$ подвергается приращениям, соответствующим алгоритму коррекции ошибок (см. задачу 3.9):

$$\Delta\mathbf{W} = \eta(\mathbf{x}_k - \mathbf{W}\mathbf{x}_k)\mathbf{x}_k, \tag{14.77}$$

где η — параметр скорости обучения. Целью обучения является построение такой архитектуры, чтобы для множества возбуждений $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K$ система вела себя как линейный ассоциатор:

$$\mathbf{W}\mathbf{x}_k = \mathbf{x}_k, k = 1, 2, \ldots, K. \tag{14.78}$$

Алгоритм коррекции ошибок (14.77) аппроксимирует идеальное условие (14.78) в смысле минимальной среднеквадратической ошибки (LMS error). В результате такого процесса обучения линейный ассоциатор формирует конкретный набор собственных векторов (определяемый векторами обучения) с собственными значениями, равными единице.

Для кластеризации сигналов радара модель BSB использует линейный ассоциатор, обучаемый методом коррекции ошибок для построения матрицы весов $\mathbf{W}$ и осуществления следующего вычисления [56]:

$$\mathbf{x}(n+1) = \varphi(\gamma\mathbf{x}(n) + \beta\mathbf{W}\mathbf{x}(n) + \delta\mathbf{x}(0)), \tag{14.79}$$

что слегка отличается от версии алгоритма BSB, представленного формулами (14.63) и (14.64). Это отличие заметно в двух аспектах.

- Константа уменьшения (decay) введена в первом слагаемом, $\gamma\mathbf{x}(n)$, для указания на некоторое уменьшение текущего состояния. Предполагая, что γ — положительная константа, меньшая единицы, ошибки в конечном счете можно свести к нулю.
- Третье слагаемое $\delta\mathbf{x}(0)$ введено для сохранения влияния вектора начального состояния $\mathbf{x}(0)$, что ограничивает возможные состояния модели BSB.

Повторение итераций модели BSB приводит к доминированию действия собственных векторов матрицы $\mathbf{W}$ с наибольшими положительными собственными значениями. Таким образом, векторы $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K$ обучаются линейным ассоциатором. Способность к кластеризации в модели BSB появилась в основном за счет ассоциации связанных с сигналом собственных векторов с большими собственными значениями, которая усиливается за счет положительной обратной связи. Таким образом, после некоторого количества итераций эти состояния модели начинают доминировать. С другой стороны, связанные с шумом собственные векторы обычно ассоциируются с малыми собственными значениями. Таким образом, их влияние на состояние модели BSB существенно уменьшается. Все это поддерживает отношение сигнал/шум на достаточно высокой отметке.

В разрезе задачи радарного наблюдения детальное описание излучателей, работающих в окрестности, заранее не известно. В доли секунды для обработки обычно поступают сотни тысяч импульсов радара, поэтому недостатка в данных в этой задаче нет. Вся сложность состоит в том, как придать этим данным смысл. Модель BSB может помочь в этой задаче, обучая микроволновую структуру среды радара с помощью встроенного свойства кластеризации. Кластеры формируются в окрестности точечных аттракторов, представляющих конкретные излучатели (источники сигнала). Таким образом, модель BSB может идентифицировать получаемые импульсы как произведенные конкретным излучателем.

14.11. Компьютерное моделирование 2

На рис. 14.23 показаны результаты моделирования модели BSB, состоящей из двух нейронов. Матрица $\mathbf{W}$ размерности 2×2 определена следующим образом:

$$\mathbf{W} = \begin{bmatrix} 0.035 & -0.005 \\ -0.005 & 0.035 \end{bmatrix}.$$

Она симметрична, положительно определена и удовлетворяет условию (14.75).

Четыре части рисунка 14.23 соответствуют четырем различным начальным состояниям $\mathbf{x}(0)$:

а) $\mathbf{x}(0) = [0.1, 0.2]^T$;

б) $\mathbf{x}(0) = [-0.2, 0.3]^T$;

в) $\mathbf{x}(0) = [-0.8, -0.4]^T$;

г) $\mathbf{x}(0) = [0.6, 0.1]^T$.

Области, показанные на этом рисунке заштрихованными, являются четырьмя бассейнами аттракции, характеризующими эту модель. На этом рисунке ясно видно, что начальное состояние модели лежит в конкретном бассейне аттракции, что определяет направление движения матрицы весов $\mathbf{W}(n)$ при увеличении количества итераций n,

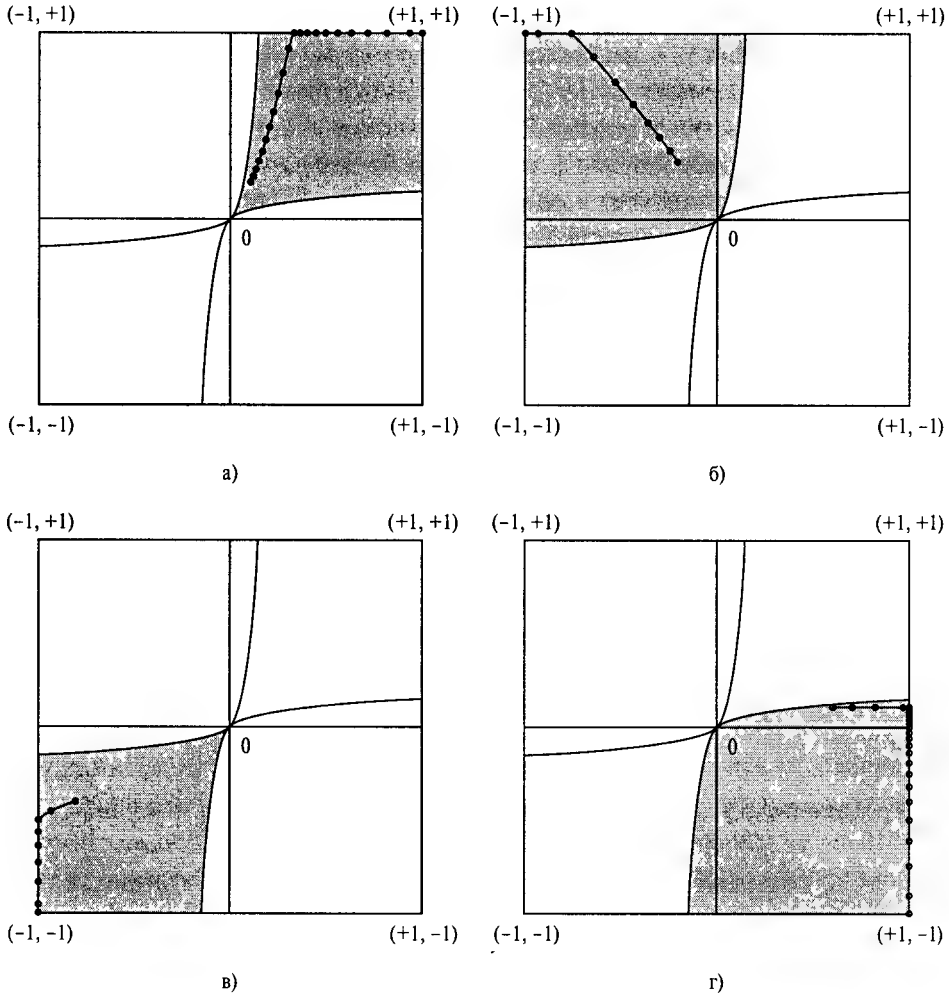


Рис. 14.23. Траектории при компьютерном моделировании модели BSB. Результаты, показанные в четырех частях рисунка, соответствуют различным начальным состояниям

до тех пор, пока состояние сети $\mathbf{x}(n)$ не достигнет фиксированного точечного аттрактора (т.е. одного из углов квадрата размера два на два), принадлежащего данному бассейну аттракции. Особый интерес для нас представляет траектория, показанная на рис. 14.23, *г*: начальное состояние $\mathbf{x}(0)$ находится в первом квадранте, а траектория заканчивается в углу четвертого (точка (+1, –1)), так как именно в бассейне этого аттрактора точки находилось начальное состояние.

14.12. Странные аттракторы и хаос

Вплоть до этого места наша дискуссия о нейродинамике фокусировала внимание читателя только на одном типе поведения динамических систем, характеризующемся наличием фиксированных точечных аттракторов. В этом разделе мы рассмотрим еще один класс аттракторов, называемых странными (strange), которые характеризуют нелинейные динамические системы с порядком выше второго.

Странные аттракторы имеют в высшей мере сложное поведение. Особенно интересным делает изучение странных аттракторов и хаоса тот факт, что рассматриваемая система является детерминированной (т.е. руководствуется фиксированными правилами) и в то же время при наличии только нескольких степеней свободы ведет себя настолько сложным образом, что внешне кажется, что это поведение носит чисто случайный характер. И в самом деле, случайность является фундаментальной характеристикой таких систем в том смысле, что статистика второго порядка хаотических временных рядов говорит именно об этом. Однако, в отличие от истинно случайных явлений, случайность, существующая в хаотических системах, не исчезает за счет сбора большего количества информации! В принципе будущее поведение хаотических систем полностью определяется их прошлым, однако на практике любая неопределенность в выборе начальных состояний, какими бы малыми они ни были, ведет к ее экспоненциальному росту с течением времени. Следовательно, несмотря на то, что динамика хаотических систем прогнозируема на короткий период времени, ее невозможно предсказать на длительный срок. Таким образом, хаотические временные ряды парадоксальны в том смысле, что их генерация осуществляется детерминированной динамической системой, но они имеют внешний вид случайных. Именно на этом свойстве явления хаоса Лоренц акцентировал внимание, открыв аттракторы, названные его именем [674].

В нелинейной динамической системе, когда орбиты аттракторов в окрестности начальных состояний стремятся отдалиться с течением времени, говорится о наличии странных аттракторов (strange attractor), а сама система называется хаотической (chaotic). Другими словами, именно фундаментальное свойство чувствительности к начальным состояниям (sensitive dependence on initial condition) делает эти аттракторы странными. Чувствительность в контексте нашей задачи означает следующее: если две идентичные системы начинают свое движение из слегка отличающихся начальных состояний (а именно $\mathbf{x}$ и $\mathbf{x} + \boldsymbol{\varepsilon}$, где $\boldsymbol{\varepsilon}$ — очень малая величина), то их динамические состояния в пространстве состояний будут расходиться друг от друга, при этом расстояние между ними будет увеличиться в среднем экспоненциально.

Инвариантные характеристики хаотической динамики

В качестве классификаторов хаотических процессов выступают два основных свойства — фрактальные измерения и экспоненты Ляпунова. Фрактальные измерения (fractal dimension) характеризуются геометрической структурой странных аттракторов. Термин "фрактальность" был введен в [702]. В отличие от целочисленных измерений (которые используются для двухмерных поверхностей и трехмерных объектов) фрактальные не принимают целых значений. Как и экспоненты Ляпунова, они описывают перемещение орбит аттракторов при эволюции динамики. Эти две инвариантные характеристики хаотической динамики будут подробно описаны ниже. Термин "инвариантность" подчеркивает тот факт, что как экспоненты Ляпунова, так и фрактальные измерения любого хаотического процесса остаются неизменными при гладких нелинейных изменениях его системы координат [2].

Фрактальные измерения

Рассмотрим странный аттрактор, динамика которого в d-мерном пространстве состояний описывается следующим образом:

$$\mathbf{x}(n+1) = \mathbf{F}(\mathbf{x}(n)), n = 0, 1, 2, \ldots, \tag{14.80}$$

что является дискретной версией выражения (14.2). Этот факт можно увидеть, приняв $t = n\Delta t$, где Δt — период дискретизации. Предполагая достаточную малость величины Δt, можно записать:

$$\frac{d}{dt}\mathbf{x}(t) = \frac{1}{\Delta t}[\mathbf{x}(n\Delta t + \Delta t) - \mathbf{x}(n\Delta t)].$$

Таким образом, сформулировать дискретную версию выражения (14.2) можно следующим образом:

$$\frac{1}{\Delta t}[\mathbf{x}(n\Delta t + \Delta t) - \mathbf{x}(n\Delta t)] = \mathbf{F}(\mathbf{x}(n\Delta t)) \text{ для малых } \Delta t.$$

Принимая для упрощения выкладок $\Delta t = 1$ и переставляя слагаемые, получим выражение

$$\mathbf{x}(n+1) = \mathbf{x}(n) + \mathbf{F}(\mathbf{x}(n)),$$

которое можно привести к форме (14.80) простым переопределением вектор-функции $\mathbf{F}(\cdot)$.

Возвращаясь к выражению (14.80), предположим, что была построена малая сфера радиуса r вокруг некоторой точки $\mathbf{y}$, лежащей на орбите аттрактора или вблизи нее. Тогда естественное распределение (natural distribution) точек аттрактора можно определить следующим образом:

$$\rho(\mathbf{y}) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \delta(\mathbf{y} - \mathbf{x}(n)), \tag{14.81}$$

где $\delta(\cdot)$ — d-мерная дельта-функция; N — количество точек данных. Обратите внимание на изменения, связанные с использованием числа N. Естественное распределение $\rho(\mathbf{y})$ выступает в роли странного аттрактора, который аналогичен функции плотности вероятности для случайной переменной. Следовательно, можно определить инвариант $\bar{f}$ по отношению к функции $f(\mathbf{y})$, при условии, что эволюция динамики описывается следующим многомерным интегралом:

$$\bar{f} = \int_{-\infty}^{\infty} f(\mathbf{y})\rho(\mathbf{y})d\mathbf{y}. \tag{14.82}$$

Рассматриваемая функция $f(\mathbf{y})$ дает меру того, как количество точек в малой сфере изменяется при устремлении радиуса r сферы к нулю. Вспоминая, что объем, ограниченный d-мерной сферой, пропорционален r^d, можно составить представление об измерении аттрактора как о поведении плотности точек на аттракторе на малых расстояниях в пространстве состояний.

Евклидово расстояние между центром $\mathbf{y}$ сферы и точкой $\mathbf{x}(n)$ на шаге дискретизации n равно $||\mathbf{y} - \mathbf{x}(n)||$. Исходя из этого, точка $\mathbf{x}(n)$ лежит внутри сферы радиуса r, если

$$||\mathbf{y} - \mathbf{x}(n)|| < r,$$

или, эквивалентно:

$$r - ||\mathbf{y} - \mathbf{x}(n)|| > 0.$$

Таким образом, функцию $f(\mathbf{x})$ в описываемой здесь ситуации можно записать в общем виде:

$$f(\mathbf{x}) = \left(\frac{1}{N-1} \sum_{k=1, k\neq n}^{N} \theta(r - ||\mathbf{y} - \mathbf{x}(k)||) \right)^{q-1}, \tag{14.83}$$

где q — целое число; $\theta(\cdot)$ — функция Хэвисайда (Heaviside), определяемая следующим образом:

$$\theta(z) = \begin{cases} 1, & z > 0, \\ 0, & z < 0. \end{cases}$$

Подставляя выражения (14.81) и (14.83) в (14.82), получим новую функцию $C(q, r)$, которая зависит от q и r следующим образом:

$$C(q,r) = \int_{-\infty}^{\infty} \left(\frac{1}{N-1} \sum_{k=1, k \neq n}^{N} \theta(r - \|\mathbf{y} - \mathbf{x}(k)\|) \right)^{q-1} \left(\frac{1}{N} \sum_{n=1}^{N} \delta(\mathbf{y} - \mathbf{x}(n)) \right) d\mathbf{y}.$$

Используя свойство сортировки дельта-функции

$$\int_{-\infty}^{\infty} g(\mathbf{y}) \delta(\mathbf{y} - \mathbf{x}(n)) d\mathbf{y} = g(\mathbf{x}(n))$$

для некоторой функции $g(\cdot)$ и изменяя порядок суммирования, можно переопределить функцию $C(q, r)$ в следующем виде:

$$C(q,r) = \frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{N-1} \sum_{k=1, k \neq n}^{N} \theta(r - \|\mathbf{x}(n) - \mathbf{x}(k)\|) \right)^{q-1}. \tag{14.84}$$

Эта функция $C(q, r)$ называется функцией корреляции[7] (correlation function) и является мерой вероятности того, что две точки, $\mathbf{x}(n)$ и $\mathbf{x}(k)$, на аттракторе находятся на расстоянии r. Количество точек данных N в определении (14.84) должно быть достаточно большим.

Сама эта функция корреляции $C(q, r)$ является инвариантом аттрактора. Тем не менее на практике общепринято фокусировать внимание на поведении $C(q, r)$ при малых r. Предельный случай описывается следующим выражением:

$$C(q,r) \simeq r^{(q-1)D_q}, \tag{14.85}$$

где предполагается существование D_q, называемого фрактальным измерением (fractal dimension) аттрактора. Взяв логарифм от обеих частей уравнения (14.85), можно

[7] Идея функции корреляции $C(q, r)$ (см. (14.84)) была известна в статистике с [880]. Однако для характеристики ложного аттрактора она использовалась только в [375]. Изначально использование функции корреляции $C(q, r)$ рассматривалось в контексте размерности корреляции $q = 2$.

формально определить D_q следующим образом:

$$D_q = \lim_{r \to 0} \frac{\log C(q,r)}{(q-1)\log r}. \tag{14.86}$$

Однако, так как обычно приходится иметь дело с конечным количеством точек данных, радиус r должен быть достаточно малым, чтобы достаточно точек поместилось внутри сферы. Для наперед заданного q фрактальное измерение D_q можно определить как наклон той части функции $C(q,r)$, которая в логарифме $log\ r$ линейна.

При $q = 2$ определение фрактального измерения D предполагает наличие простой формы, удобной для вычислений. Полученное измерение D_2 называется измерением корреляции (correlation dimension) аттрактора [375]. Измерение корреляции отражает сложность рассматриваемой динамической системы и ограничивает степени свободы, необходимые для описания системы.

Экспоненты Ляпунова

Экспоненты Ляпунова являются статистическими величинами, которые описывают неопределенность будущих состояний аттрактора. Более конкретно, они оценивают экспоненциальное расстояние, на которое соседние траектории отдалены друг от друга при движении к аттрактору. Пусть $\mathbf{x}(0)$ — начальное состояние, а $\{\mathbf{x}(n), n = 0, 1, 2, \ldots\}$ — соответствующая орбита. Рассмотрим точку на бесконечно малом удалении от начального состояния $\mathbf{x}(0)$ в направлении вектора $\mathbf{y}(0)$, тангенциального к орбите. Тогда эволюция вектора тангенса определяет эволюцию бесконечно малого смещения возмущенной орбиты $\{\mathbf{y}(n), n = 0, 1, 2, \ldots\}$ от невозмущенной $\{\mathbf{x}(n), n = 0, 1, 2, \ldots\}$. В частности, отношение $\mathbf{y}(n)/||\mathbf{y}(n)||$ определяет бесконечно малое смещение орбиты от точки $\mathbf{x}(n)$, а отношение $||\mathbf{y}(n)||/||\mathbf{y}(0)||$ — множитель, на который бесконечно малое смещение *растет* (если $||\mathbf{y}(n)|| > ||\mathbf{y}(0)||$) или уменьшается (если $||\mathbf{y}(n)|| > ||\mathbf{y}(0)||$). Для любого начального состояния $\mathbf{x}(0)$ и начального смещения $\alpha_0 = \mathbf{y}(0)/||\mathbf{y}(0)||$ экспонента Ляпунова (Lyapunov exponent) определяется следующим образом:

$$\lambda(\mathbf{x}(0), \alpha) = \lim_{n \to \infty} \frac{1}{n} \log \left(\frac{||\mathbf{y}(n)||}{||\mathbf{y}(0)||} \right). \tag{14.87}$$

Любой d-мерный хаотический процесс имеет в целом d экспонент Ляпунова, которые могут быть положительными, отрицательными и равняться нулю. Положительные экспоненты Ляпунова учитывают неустойчивость орбиты в пространстве состояний. Другими словами, положительные экспоненты Ляпунова отвечают за чувствительность хаотического процесса к начальным состояниям. С другой стороны, отрицательные экспоненты Ляпунова управляют снижением нестационарности орбиты (decay of transients in orbit). Нулевая экспонента Ляпунова отмечает тот факт, что

динамику хаоса можно описать системой нелинейных дифференциальных уравнений. Это значит, что хаотический процесс является *потоком* (flow). Объем в d-мерном пространстве состояний ведет себя как $\exp(L(\lambda_1 + \lambda_2 + \ldots + \lambda_d))$, где L — количество шагов времени в будущем. Отсюда следует, что, для того, чтобы процесс был диссипативным (dissipative), сумма экспонент Ляпунова должна быть отрицательной. Это необходимое условие того, чтобы объем в пространстве состояний уменьшался со временем, что является условием физической реализуемости.

Измерение Ляпунова

Для заданного спектра Ляпунова $\lambda_1, \lambda_2, \ldots, \lambda_d$ в [539] было введено понятие измерения Ляпунова (Lyapunov dimension) для странного аттрактора:

$$D_L = K + \frac{\sum_{i=1}^{K} \lambda_i}{|\lambda_{K+1}|}, \tag{14.88}$$

где K — целое число, удовлетворяющее двум условиям:

$$\sum_{i=1}^{K} \lambda_i > 0 \text{ и } \sum_{i=1}^{K+1} \lambda_i < 0.$$

При нормальных условиях измерение Ляпунова D_L имеет приблизительно тот же размер, что и измерение корреляции D_2. Это важное свойство любого хаотического процесса. Это значит, что несмотря на то, что измерения корреляции и Ляпунова определяются совершенно различными способами, их значения для странного аттрактора обычно достаточно близки друг к другу.

Определение хаотического процесса

В этом разделе обсуждался хаотический процесс, однако не давалось его формальное определение. В свете того, что известно об экспонентах Ляпунова, можно принять следующее определение.

Хаотический процесс — это процесс, генерируемый нелинейной динамической системой, в которой хотя бы одна экспонента Ляпунова положительна.

Это необходимое условие для чувствительности к начальным состояниям, что является основным признаком странного аттрактора.

Самая большая экспонента Ляпунова также определяет горизонт прогнозирования хаотического процесса. В частности, кратковременная прогнозируемость хаотического процесса приблизительно равна величине, обратной к максимальной экспоненте Ляпунова [2].

14.13. Динамическое восстановление

Динамическое восстановление можно определить как отображение, которое воспроизводит модель неизвестной динамической системы размерности m. Интерес представляет динамическое моделирование временных рядов, генерируемых системой, о которой известно, что она хаотична. Другими словами, для заданного временного ряда $\{y(n)\}_{n=1}^{N}$ необходимо построить некоторую модель, которая вобрала бы в себя рассматриваемую динамику, ответственную за генерацию наблюдения $y(n)$. Как уже говорилось в предыдущем разделе, N — это размерность множества примеров. Основной целью динамического восстановления является придание физического смысла таким временным рядам без необходимости математических знаний о рассматриваемой динамике. Обычно рассматриваемая система чересчур сложна, чтобы ее можно было охарактеризовать математическими терминами. Нам доступна только информация, содержащаяся во временных рядах, полученных из измерений одного из наблюдений системы.

Фундаментальным результатом теории динамического восстановления[8] является геометрическая теорема, называемая, согласно [1040], теоремой вложения с задержкой (delay-embedding theorem). Такенс рассматривал ситуацию, в которой отсутствует шум, фокусируя внимание на координатных отображениях с задержкой (delay coordinate maps) или моделях прогнозирования, которые конструируются из временных рядов, представляющих собой наблюдения динамической системы. В частности, Такенс показал, что если динамическая система и наблюдения носят общий характер, то координатное отображение с задержкой из d-мерного гладкого компактного множества в $\Re^{2d+1}$ является диффеоморфизмом (diffeomorphism) этого многообразия, где d — размерность пространства состояний данной динамической системы. (Диффеоморфизм рассматривается в следующей главе.)

Для интерпретации теоремы Такенса в терминах обработки сигнала вначале рассмотрим неизвестную динамическую систему, эволюция которой в дискретном времени описывается нелинейным разностным уравнением:

$$\mathbf{x}(n+1) = \mathbf{F}(\mathbf{x}(n)), \tag{14.89}$$

где $\mathbf{x}(n)$ — d-мерный вектор состояний системы в момент времени n; $\mathbf{F}(\cdot)$ — некото-

[8] Построение динамики, использующее независимые координаты из временных рядов, впервые было проведено в [809]. Однако в этой работе не содержалось никаких доказательств и вместо вложений с задержкой по времени использовались "производные" вложения. В 1981 году Такенс опубликовал математически глубокую работу, посвященную вложениям с задержкой по времени, которые применялись к аттракторам, подобным поверхностям или тору. В том же году была опубликована работа, посвященная тому же вопросу [703]. Нематематикам будет сложно читать работу Такенса, и еще сложнее работу Мане. Идея координатного отображения с задержкой (delay coordinate mapping) была обогащена [935]. Подход, предпринятый в этой работе, объединяет и расширяет результаты предшественников [1040], [1135].

рая вектор-функция. Здесь предполагается, что период дискретизации нормализован к единице. Пусть временной ряд $\{y(n)\}$, наблюдаемый на выходе системы, определяется в терминах вектора состояний $\mathbf{x}(n)$ следующим образом:

$$y(n) = g(\mathbf{x}(n) + v(n), \tag{14.90}$$

где $g(\cdot)$ — скалярная функция; $v(n)$ — аддитивный шум. Здесь аддитивный шум добавлен для того, чтобы учесть несовершенство и неточность методов измерения наблюдения $y(n)$. Уравнения (14.89) и (14.90) описывают поведение динамической системы в пространстве состояний. Согласно теореме Такенса, геометрическая структура динамики этой системы, зависящей от многих переменных, может быть восстановлена (unfold) на основе наблюдений $y(n)$ с нулевым шумом ($v(n) = 0$) в D-мерном пространстве, построенном на основе нового вектора

$$\mathbf{y}_R(n) = [y(n), y(n-\tau), \ldots, y(n-(D-1)\tau)]^T, \tag{14.91}$$

где τ — некоторое положительное целое, называемое нормализованной задержкой вложения (normalized embedding delay). Это значит, что для заданного наблюдаемого значения $y(n)$ в дискретный момент времени n, которое относится к одному компоненту неизвестной динамической системы, динамическое восстановление возможно с использованием D-мерного вектора $\mathbf{y}_R(n)$ ($D \geq 2d+1$), где d — размерность пространства состояний данной системы. Здесь и далее это утверждение будем называть теоремой вложения с задержкой (delay embedding theorem). Условие $D \geq 2d+1$ является достаточным, но не является необходимым для динамического восстановления. Процедура поиска подходящего D называется вложением, или *вставкой* (embedding), а минимальное целое число D, при котором достигается динамическое восстановление, называется измерением вложения (embedding dimension) и обозначается как D_E.

Теорема вложения с задержкой имеет большое применение. Эволюция точек $\mathbf{y}_R(n) \rightarrow \mathbf{y}_R(n+1)$ в реконструируемом пространстве соответствует эволюции точек $\mathbf{x}(n) \rightarrow \mathbf{x}(n+1)$ в исходном пространстве состояний. Это значит, что большинство свойств ненаблюдаемого вектора состояний $\mathbf{x}(n)$ воспроизводится без неоднозначности в пространстве восстановления, определяемом вектором $\mathbf{y}_R(n)$. Однако, для того, чтобы этот важный результат был достижим, требуются достоверные оценки (reliable estimate) измерения вложения D_E и нормализованное запаздывание вложения τ.

- Достаточное условие $D \geq 2d+1$ делает возможным восстановление (undo) пересечений орбит аттракторов со своими же орбитами, которые возникают в результате проекций этих орбит на измерения более низкого порядка. Измерение вложения D_E может быть меньше величины $2d+1$. Рекомендуется оценивать D_E непосредственно на основе данных наблюдений. Достоверным методом оценки

D_E является метод ложных ближайших соседей (method of false nearest neighbors), описанный в [2]. В этом методе симметрично обследуются точки данных и их соседи для измерения $d = 1$, затем для d =2 и т.д. Таким образом, можно установить состояние, при котором очевидные соседи станут различимыми при добавлении к вектору восстановления $\mathbf{y}_R(n)$ большего числа элементов, и получить оценку D_E измерения вложения.

- К сожалению, теорема вложения с задержкой ничего не говорит о выборе нормализованной задержки вложения τ. На самом деле она разрешает использовать любое значение τ, пока доступные временные ряды бесконечно длинны. Однако на практике всегда приходится работать с данными наблюдений конечной длины N. Правильные установки при выборе τ предполагают, что эта величина должна быть достаточно большой, чтобы $y(n)$ и $y(n-\tau)$ были достаточно независимы друг от друга для того, чтобы служить координатами пространства восстановления, но не настолько независимыми, чтобы совершенно не иметь корреляции друг с другом. Это требование лучше всего удовлетворяется при использовании такого τ, при котором взаимная информация (mutual information) между $y(n)$ и $y(n-\tau)$ достигает своего первого минимума [308]. (О понятии взаимной информации см. в главе 10.)

Рекурсивное прогнозирование

В представленном обсуждении задача динамического восстановления может интерпретироваться как представление свойства динамики сигнала (шаг вложения) и как создание прогнозирующего отображения (шаг идентификации). Таким образом, если перейти к практическим терминам, получим следующую топологию сети для динамического моделирования.

- Структура кратковременной памяти (например, память на линиях задержки) осуществляет вложение, в то время как вектор восстановления $\mathbf{y}_R(n)$ определяется в терминах наблюдений $y(n)$ и их версий с задержкой (см. (14.91)).
- Адаптивная нелинейная система с несколькими входами и одним выходом MISO обучается как система одношагового прогнозирования (например, нейронная сеть) для поиска неизвестного отображения f: $\Re^D \rightarrow \Re^1$, определяемого следующим образом:

$$\hat{y}(n+1) = f(\mathbf{y}_R(n)). \tag{14.92}$$

Прогнозирующее отображение (14.92) занимает центральное место в динамическом программировании. Как только оно определено, эволюция $\mathbf{y}_R(n) \rightarrow \mathbf{y}_R(n+1)$ становится известной, что, в свою очередь, определяет неизвестную эволюцию $\mathbf{x}(n) \rightarrow \mathbf{x}(n+1)$.

Рис. 14.24. Система одношагового прогнозирования, используемая при итеративном прогнозировании для динамического восстановления хаотических процессов

В настоящее время не существует какой-либо строгой теории, которая помогла бы решить, успешно ли система нелинейного прогнозирования идентифицирует неизвестное отображение f. В линейной системе прогнозирования минимизация среднеквадратического значения ошибки прогнозирования приводит к определению точной модели. Однако при работе с хаотичными временными рядами все обстоит иначе. Две траектории в одном и том же аттракторе могут значительно отличаться для разных образов. Исходя из этого, минимизация среднеквадратического значения ошибки прогнозирования является необходимым, но не достаточным условием для определения успешности отображения.

Динамические инварианты (измерение корреляции и экспоненты Ляпунова) определяют глобальные свойства аттрактора, так что их можно использовать в динамическом моделировании в качестве меры. Таким образом, прагматический подход к тестированию динамической модели предполагает диссипативность (рассеивание) точек ложных аттракторов и замыкание ее выхода на вход, подобно автономной системе, показанной на рис. 14.24. Такая операция называется итеративным или рекурсивным прогнозированием (iterated или recursive prediction). После того как выполнена инициализация, выход этой автономной системы представляет собой реализацию процесса динамического восстановления. Естественно, это предполагает в первую очередь наличие грамотной конструкции этой системы прогнозирования.

Считается, что динамическое восстановление, выполненное автономной системой на рис. 14.24, является успешным, если выполняются следующие условия [441].

1. Условие на *краткосрочное поведение* (short-term behavior). После того как выполнена инициализация, временной ряд $\{\hat{y}(n)\}$ на рис. 14.24 должен быть достаточно близок к исходному ряду $\{y(n)\}$ в течение периода времени, в среднем равному горизонту прогнозирования, определяемому из спектра Ляпунова этого процесса.
2. Условие на *долгосрочное поведение* (long-term behavior). Динамические инварианты, вычисленные для реконструированного временного ряда $\{\hat{y}(n)\}$, должны быть близки к соответствующим инвариантам исходного временного ряда $\{y(n)\}$.

Для измерения долгосрочного поведения восстановленной динамики требуется оценить степень корреляции как меру сложности аттрактора и спектр Ляпунова для оценки чувствительности к начальным состояниям и для оценки измерения Ляпунова (см. (14.88)). Измерение Ляпунова должно быть достаточно близко к измерению корреляции.

Две возможные формулировки рекурсивного прогнозирования

Вектор восстановления $\mathbf{y}_R(n)$, определяемый формулой (14.91), имеет размерность D_E, если размерность D установлена равной размерности вложения D_E. Размер памяти на линиях задержки, требуемой для такого вложения, составляет τD_E. Однако память на линиях задержки требуется для обеспечения только D_E выходов (это размерность пространства восстановления), т.е. используем τ равно отстоящих друг от друга отводов, представляющих разреженные связи (sparse connections).

В качестве альтернативы можно определить вектор восстановления $\mathbf{y}_R(n)$ как полный m-мерный вектор следующего вида:

$$\mathbf{y}_R(n) = [y(n), y(n-1), \ldots, y(n-m+1)]^T, \tag{14.93}$$

где m — целое число, удовлетворяющее условию

$$m \geq D_E \tau. \tag{14.94}$$

Эта вторая формулировка вектора восстановления $\mathbf{y}_R(n)$ обеспечивает модель прогнозирования бо́льшим количеством информации, чем (14.91), и, таким образом, приводит к более точному динамическому восстановлению. Однако обе эти формулировки обладают одним общим свойством: их композиции однозначно определяются знанием измерения вложения D_E. В любом случае лучше всего использовать минимальное из возможных значений D (а именно D_E) для минимизации эффекта, производимого на качество динамического восстановления аддитивным шумом $v(n)$.

Динамическое восстановление как плохо обусловленная задача фильтрации

Задача динамического восстановления на самом деле является плохо обусловленной обратной задачей (ill-posed inverse problem). И этому имеется несколько причин. (Условия хорошей обусловленности обратных задач см. в главе 5.) Во-первых, по некоторой неизвестной причине может быть нарушено условие существования. Во-вторых, для единственности восстановления нелинейной динамики в доступных наблюдениях может оказаться недостаточно информации. В-третьих, неизбежное наличие аддитивного шума и некоторых форм неточности в наблюдаемых временны́х рядах добавляет в динамическое восстановление неопределенность. В частности, если уровень шума достаточно велик, возможно нарушение условия непрерывности. Как же добиться хорошей обусловленности в задаче динамического восстановления? Ответ на этот вопрос лежит во включении некоторой формы априорных знаний об отображении вход-выход. Другими словами, на модели прогнозирования, предназначенные для решения задач динамического прогнозирования, должны налагаться

некоторые формы ограничений (например, на гладкость преобразования). Одним из эффективных способов удовлетворения этому требованию является использование теории регуляризации Тихонова (см. главу 5).

Еще одним вопросом, на котором следует остановиться, является способность модели прогнозирования с достаточной точностью решать обратную задачу. В этом контексте подходящим решением для построения модели прогнозирования будет использование нейронных сетей. В частности, свойство универсального аппроксиматора многослойного персептрона или сетей на основе радиальных базисных функций означает, что можно позаботиться о точности восстановления при использовании одной из этих сетей соответствующего размера. Однако по изложенным ранее причинам требуется регуляризировать решение. Теоретически для использования регуляризации пригодны как многослойный персептрон, так и сети на основе радиальных базисных функций. На практике именно в сетях на основе радиальных базисных функций теория регуляризации была математически включена как составная часть их конструкции. В соответствии с этим при последующем компьютерном моделировании сосредоточим внимание на сети на основе регуляризованных радиальных базисных функций (см. главу 5) как основе решения задачи динамического восстановления.

14.14. Компьютерное моделирование 3

Чтобы проиллюстрировать идею динамического восстановления, рассмотрим систему трех обычных дифференциальных уравнений, выделенных Лоренцом из аппроксимации Галеркина с помощью уравнений в частных производных термальной конвекции в нижних слоях атмосферы. Эта задача послужит в качестве тестовой системы уравнений для проверки идей нелинейной динамики. Уравнения аттрактора Лоренца имеют следующий вид:

$$\begin{aligned}
\frac{dx(t)}{dt} &= -\sigma x(t) + \sigma y(t), \\
\frac{dy(t)}{dt} &= -x(t)z(t) + rx(t) - y(t), \\
\frac{dz(t)}{dt} &= x(t)y(t) - bz(t),
\end{aligned} \tag{14.95}$$

где σ, r и b — безразмерные параметры. Типичные значения этих параметров — $\sigma = 10$, $r = 28$ и $b = 8/3$.

На рис. 14.25 показаны результаты итеративного прогнозирования, выполненного на двух сетях RBF, имеющих 400 центров и использовавших "зашумленные" временные ряды, основанные на компоненте $x(t)$ аттрактора Лоренца. Соотношение сигнал/шум составляло +25 dB. На рис. 14.25, *а* конструкция сети регуляризирована, а на рис. 14.25, *б* — нет. Эти две части рисунка ясно демонстрируют практическую важность регуляризации. При отсутствии регуляризации решение задачи динамиче-

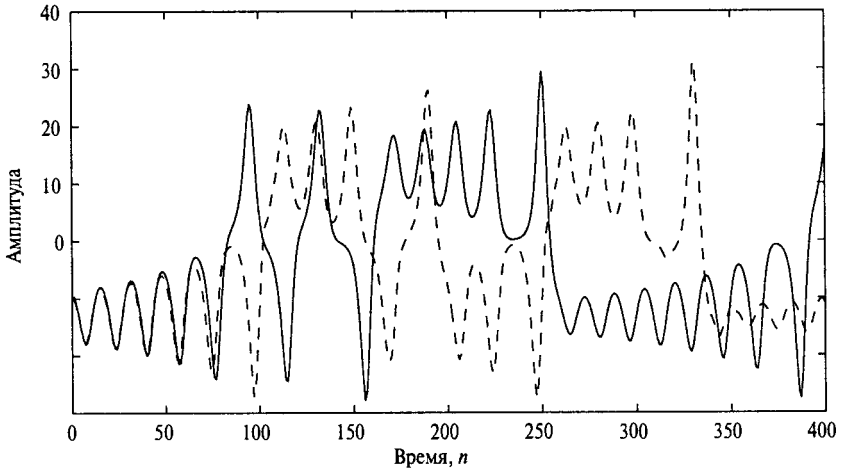
ского восстановления, представленное на рис. 14.25, *б*, неприемлемо, поскольку оно не способно аппроксимировать истинную траекторию аттрактора Лоренца. Нерегуляризированная система является всего лишь системой прогнозирования. С другой стороны, решение задачи динамического восстановления представлено на рис. 14.25, *а*. В этом решении использована регуляризированная форма сети RBF, которая обучалась динамике в том смысле, что выходной сигнал сети при итеративном прогнозировании близко аппроксимировал истинную траекторию аттрактора Лоренца на краткосрочный период времени. Это подтверждается результатами, представленными в табл. 14.5, в которой сведены данные для трех случаев.

1. Незашумленная система Лоренца.
2. Зашумленная система Лоренца с соотношением сигнал/шум, равным SNR=25dB.
3. Данные, восстановленные с использованием зашумленных временных рядов Лоренца, показаны на рис. 14.25, *а*.

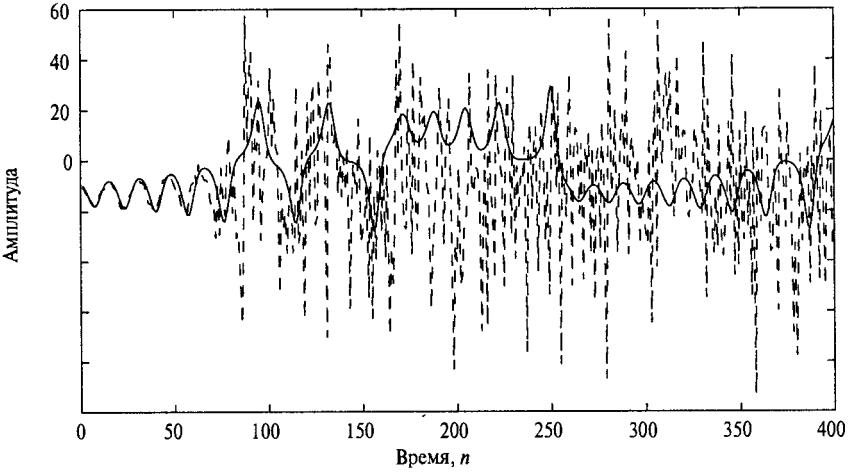
Инварианты данных, восстановленных с использованием зашумленных рядов, достаточно близки к соответствующим инвариантам незашумленных данных Лоренца. Отклонения в абсолютных значениях связаны с остаточным эффектом шума, вкравшимся в реконструированный аттрактор и в неточности процедуры оценки. На рис. 14.25 ясно показано, что динамическое моделирование — это нечто большее, нежели просто прогнозирование. На этом рисунке (как и на многих других, не представленных в этой книге) продемонстрирована "робастность" регуляризованного решения RBF по отношению к точке аттрактора, использованной для инициализации процесса итеративного прогнозирования.

Стоит обратить внимание на следующие два наблюдения, касающиеся использования регуляризации (см. рис. 14.25, *а*).

1. Краткосрочная прогнозируемость реконструированных временных рядов на рис. 14.25, *а* составляет более 60 образцов. Теоретический горизонт прогнозируемости, вычисленный из спектра Ляпунова незашумленного аттрактора Лоренца, составляет приблизительно 100 образов. Экспериментальное отклонение от горизонта прогнозируемости незашумленного аттрактора Лоренца вызвано в основном наличием шума в фактических данных, использованных для динамического восстановления. Теоретический горизонт прогнозируемости, вычисленный на основе реконструированных данных, составил 61 (табл. 14.5), что достаточно близко к экспериментально наблюдаемому значению краткосрочной прогнозируемости.
2. Как только период краткосрочной прогнозируемости превышен, реконструированные временные ряды на рис. 14.25, *а* начинают отклоняться от незашумленной реализации фактического аттрактора Лоренца. Это в основном объясняется хаотической динамикой, а именно чувствительностью к начальным состояниям. Как уже говорилось ранее, чувствительность к начальным состояниям является характерной чертой хаоса.



а)



б)

Рис. 14.25. Регуляризированное итеративное прогнозирование (N=400, m=20) на данных Лоренца при SNR=+25dB (а); нерегуляризированное итеративное прогнозирование ($N = 400$, $m = 20$) на данных Лоренца при SNR=+25dB (б). В обеих частях рисунка непрерывная кривая представляет собой фактический хаотический сигнал, а пунктирная кривая — реконструированный сигнал

Примечание. Все экспоненты Ляпунова представлены в натах в секунду. *Нат* (nat) — это естественная единица измерения информации, описанная в главе 10. В случае (б) эффект шума выразился в увеличении размера спектра Ляпунова, а также в увеличении количества и амплитуды положительных экспонент Ляпунова.

ТАБЛИЦА 14.5. Параметры моделирования динамического восстановления, использующего систему Лоренца

(а) Система Лоренца без учета шума
Количество использованных образов: 35000
1. Нормализованная задержка вложения, τ =4
2. Измерение вложения, D_E =3
3. Экспоненты Ляпунова:
$$\lambda_1 = 1.5697;$$
$$\lambda_2 = -0.0314;$$
$$\lambda_3 = -22.3054$$
4. Горизонт прогнозируемости $\approx$ 100 образцов

(б) Зашумленная система Лоренца: +25 dB SNR
Количество использованных образов: 35000
1. Нормализованная задержка вложения, τ =4
2. Измерение вложения, D_E =5
3. Экспоненты Ляпунова:
$$\lambda_1 = 13.2689;$$
$$\lambda_2 = 5.8562;$$
$$\lambda_3 = -3.1447;$$
$$\lambda_4 = -18.0082;$$
$$\lambda_5 = -47.0572$$
4. Горизонт прогнозируемости $\simeq$ 12 образцов

(в) Система, восстановленная с использованием зашумленных данных Лоренца (см. рис. 14.25, а)
Количество сгенерированных образов: 35000
1. Нормализованная задержка вложения, τ =4
2. Измерение вложения, D_E =3
3. Экспоненты Ляпунова:
$$\lambda_1 = 2.5655;$$
$$\lambda_2 = -0.6275;$$
$$\lambda_3 = -15.0342$$
4. Горизонт прогнозируемости $\simeq$ 61 образец

Выбор параметров m и λ

Размер входного слоя m определяется по формуле (14.94). Как уже говорилось ранее, рекомендуется использовать наименьшее допустимое значение m в соответствии со знаком равенства, минимизирующим эффект шума при динамическом восстановлении.

Оцененное значение нормализованной задержки вложения τ не зависит от наличия шума при высоких значениях отношения сигнал/шум. В противоположность этому наличие шума оказывает ощутимое воздействие на оценочное значение измерения вложения D_E, что интуитивно понятно. Например, для незашумленного аттрактора Лоренца измерение корреляции составило 2,01. Таким образом, можно выбрать такое измерение вложения D_E, которое подтверждено методом ложных ближайших соседей (false nearest neighbors). Нормализованная задержка вложения составила τ =4. Таким образом, при использовании выражения (14.94) со знаком равенства получим для динамического восстановления значение m = 12. С другой стороны, для зашумленного аттрактора Лоренца при SNR=+25dB использование метода ложных ближайших соседей дает значение D_E =5, а использование метода взаимной информации приводит к значению τ =4. Подставляя эти оцененные значения в (14.94) со знаком равенства, получим для зашумленного динамического восстановления значение m =20 (см. рис. 14.25). В табл. 14.5 представлены значения задержки вложения τ и измерения вложения D_E для всех трех случаев.

Что же касается параметра регуляризации λ, использованного на рис. 14.25, *а*, то он определялся из данных обучения с помощью обобщенной перекрестной проверки (Generalized Cross-Validation — GCV) (см. главу 5). Полученное таким образом значение λ варьировалось от минимального значения 10^{-14} до максимального — 10^{-2}, в зависимости от данных.

14.15. Резюме и обсуждение

Большая часть материала, представленного в настоящей главе, посвящена модели Хопфилда и модели BSB (brain-state-in-box). Это примеры ассоциативной памяти, базирующейся на нейродинамике. Эти две модели обладают следующими общими характеристиками.

- Они используют положительные обратные связи.
- Они имеют функцию энергии (Ляпунова), и рассматриваемая динамика стремится последовательно ее минимизировать.
- Они используют постулат обучения Хебба и обучаются с помощью самоорганизации.
- Они способны выполнять вычисления, используя динамику аттракторов.

Естественно, они отличаются друг от друга областью применения.

Модель BSB имеет внутреннюю способность к кластеризации, которую можно успешно использовать для представления данных и формирования понятий (concept formation). Наиболее интересным применением модели BSB является, пожалуй, использование ее в качестве основного вычислительного элемента в сети сетей (network of networks) — правдоподобной модели, описывающей различные уровни системной

организации в мозге [55]. В этой модели вычислительные элементы формируют локальные сети, которые распределены в двумерном массиве (это и является причиной названия "сеть сетей"). Вместо взаимодействия столбцов только на уровне средних значений эти локальные сети сконструированы для взаимодействия с другими локальными сетями посредством образов (векторов) активности (activity pattern). На месте синаптических весов между нейронами, присутствующих в обычных сетях, в них находится множество матриц взаимодействия, которые описывают связи между аттракторами двух локальных сетей. Локальные сети формируют кластеры и уровни, основанные на их взаимных связях, в результате чего анатомическая связность (anatomical connectivity) разрежается. Это значит, что локальные сети более тесно связаны внутри кластеров, чем между кластерами. Тем не менее функциональная связность (functional connectivity) между кластерами имеет богатую динамику, частично благодаря временной коррелированности работы локальных сетей.

В противоположность этому модель Хопфилда может использоваться для решения следующих вычислительных задач.

1. Реализация ассоциативной памяти, которая включает в себя восстановление сохраненных образов при предъявлении памяти неполных или зашумленных их версий. Для этого приложения, как правило, используется "дискретная" модель Хопфилда, которая основана на нейроне Мак-Каллока–Питца (т.е. на нейроне, использующем жестко ограниченную функцию активации). Рассматривая эту модель в вычислительном контексте, можно сказать, что эта память имеет довольно тривиальную конструкцию. Тем не менее модель Хопфилда ассоциативной памяти является важной, так как она объясняет связь между динамикой и вычислениями совершенно новаторским способом. В частности, модель Хопфилда объясняет следующие свойства, имеющие нейробиологическую релевантность.
   - Динамика этой модели определяется большим количеством точечных аттракторов в пространстве состояний большой размерности.
   - Рассматриваемый точечный аттрактор (т.е. ячейка фундаментальной памяти) может быть найден с помощью простой инициализации модели неточным описанием места расположения этого аттрактора, После этого динамика сама переведет состояние модели к ближайшему точечному аттрактору.
   - Обучение (т.е. вычисление свободных параметров модели) происходит в соответствии с постулатом обучения Хебба. Более того, этот механизм обучения позволяет добавлять в модель новые точки аттракторов, если это необходимо.
2. Комбинаторные задачи оптимизации (combinatorial optimization problem), которые считаются в математике самыми сложными. Этот класс задач оптимизации включает в себя известную задачу коммивояжера (Travelling Salesman Problem — TSP), считающуюся классической. При заданном расположении известного числа городов (предполагается, что они лежат на плоскости) стоит задача поиска

кратчайшего маршрута обхода всех городов, который начинается и заканчивается в одном и том же городе. Задача TSP просто формулируется, но крайне сложно решается, поскольку не существует известного метода поиска оптимального маршрута, ищущего все возможные маршруты и затем выбирающего кратчайший из них. Эта задача называется NP-полной (NP-complete) [474]. В новаторской работе [482] было продемонстрировано, как аналоговая сеть, основанная на системе дифференциальных уравнений первого порядка (14.20), может использоваться для представления решения задачи TSP. В частности, синаптические веса этой сети определялись расстоянием между отдельными городами, а оптимальное решение являлось фиксированной точкой нейродинамических уравнений (14.20). Здесь и сосредоточены сложности, связанные с "отображением" комбинаторных задач оптимизации на непрерывные (аналоговые) сети Хопфилда. Эта сеть стремится минимизировать одну функцию энергии (Ляпунова), в то время как типичная комбинаторная задача оптимизации требует оптимизации некоторой целевой функции при наличии ряда жестких ограничений [340]. Если нарушается какое-либо из этих ограничений, решение считается неверным. Первые процедуры такого отображения основывались на функции Ляпунова, сконструированной специальным методом, обычно использующим по одному слагаемому для каждого условия:

$$E = E^{\text{опт.}} + c_1 E_1^{\text{огр.}} + c_2 E_2^{\text{огр.}} + \ldots . \tag{14.96}$$

Первое слагаемое, $E^{\text{опт.}}$, является минимизируемой целевой функцией (например, длиной маршрута в задаче TSP). Эта функция определяется рассматриваемой задачей. Оставшиеся слагаемые, $E_1^{\text{огр.}}, E_2^{\text{огр.}}, \ldots$, представляют собой штрафные функции, минимизация которых удовлетворяет ограничениям. Скаляры $c_1, c_2, \ldots$ являются постоянными весами, которые назначены соответствующим штрафным функциям $E_1^{\text{огр.}}, E_2^{\text{огр.}}, \ldots$. Они обычно определяются методом проб и ошибок. К сожалению, многочисленные слагаемые в функции Ляпунова (14.96) имеют тенденцию вытеснять друг друга, и успех сети Хопфилда в огромной мере зависит от относительных значений $c_1, c_2, \ldots$ [340]. Учитывая это, неудивительно, что такая сеть зачастую производит большое количество неверных решений [63], [1160]. В [340] рассматривалось множество основных вопросов, касающихся использования непрерывных сетей Хопфилда как инструмента разрешения комбинаторных задач оптимизации; основные ее выводы приведены ниже.

- Для заданной задачи комбинаторной оптимизации, представленной в терминах квадратичного 0-1 программирования, равно как и для коммивояжера, существует простой метод программирования сети, приводящий к ее решению, при этом найденное решение не будет нарушать ни одно из ограничений задачи.

- На основании результатов теории сложности (complexity theory) и математического программирования было показано, что за исключением случая, когда ограничения задачи имеют особые свойства, определяющие интегральный политоп или многогранник, невозможно заставить сеть сойтись к корректному, интерпретируемому решению. В геометрических терминах политоп, т.е. ограниченный многогранник, называется интегральным политопом, если все его грани являются точками 0–1. Даже когда речь идет об интегральном политопе, если целевая функция $E^{\text{опт.}}$ является квадратичной, то задача является NP-полной и нет никакой гарантии, что сеть найдет оптимальное решение. К классу этих задач относится и задача коммивояжера. Однако допустимое решение все равно может быть найдено, и в зависимости от природы используемого процесса спуска к этому решению существует шанс, что это решение окажется достоверным.

Модель Хопфилда, которая рассматривалась в настоящей главе, использовала симметричные связи между своими нейронами. Динамика такой структуры аналогична динамике градиентного спуска, что гарантирует сходимость к фиксированной точке. Однако динамика мозга отличается от динамики модели Хопфилда в двух основных вопросах.

- Взаимосвязи между нейронами мозга являются ассимметричными.
- В мозге наблюдается осциллирующее и сложное непериодическое поведение.

Естественно, причиной этого являются особые свойства мозга, которые вызвали незатухающий интерес при изучении ассимметричных сетей[9], предшествовавших модели Хопфилда.

Если отказаться от требования симметрии, следующей простейшей моделью является сеть возбуждения-торможения (excitatory-inhibitory network), нейроны которой разбиваются на две категории: с возбуждающими и тормозящими выходами. Синаптические связи между двумя этими популяциями являются ассимметричными, в то время как связи в каждой из популяций симметричны. В [966] рассматривалась динамика именно такой сети. Представленный в этой работе анализ использует сходство между динамикой сети возбуждения-торможения и динамикой градиентного спуска — градиентного подъема (gradient descent — gradient ascent dynamics), в которой уравнения движения для некоторых переменных состояния имели форму градиентного спуска, а для остальных — форму градиентного подъема. Следовательно, в отличие от динамики градиентного спуска, которая характеризует сеть Хопфилда, динамика модели, предложенной в этой работе, может сходиться к некоторой фиксированной

[9] Трактовка биологических нейронных сетей как нелинейных динамических систем, обладающих осциллирующим поведением, имеет долгую историю [30], [31], [34], [176], [1161].

точке или предельному циклу, в зависимости от выбора параметров сети. Таким образом, антисимметричная модель, предложенная в [966], имеет преимущество перед симметричной моделью Хопфилда.

Задачи

Динамические системы

14.1. Переформулируйте теорему Ляпунова для случая, когда вектор состояния $\mathbf{x}(0)$ является равновесным в динамической системе.

14.2. Сверьте блочные диаграммы на рис. 14.8, *а* и 14.8, *б* с уравнениями (14.18) и (14.19).

14.3. Рассмотрим нейродинамическую систему общего вида с неопределенной зависимостью от внутренних динамических параметров, внешних динамических возбудителей и переменных состояния. Эта система определяется следующей системой уравнений:

$$\frac{dx_j}{dt} = \varphi_j(\mathbf{W}, \mathbf{u}, \mathbf{x}),\ j = 1, 2, \ldots, N,$$

где матрица $\mathbf{W}$ представляет внутренние динамические параметры системы; вектор $\mathbf{u}$ — внешние динамические возбудители; $\mathbf{x}$ — вектор состояний, j-й элемент которого обозначается символом x_j. Предположим, что траектории этой системы сходятся к точечным аттракторам для значений $\mathbf{W}$, $\mathbf{u}$ и начальных состояний $\mathbf{x}(0)$, находящихся в некоторой рабочей области пространства состояний [839]. Обсудите, как описанная система может использоваться для следующих приложений.

а) Непрерывный оператор, в котором $\mathbf{u}$ — вход, а $\mathbf{x}(\infty)$ — выход.

б) Ассоциативная память, в которой $\mathbf{x}(0)$ — вход, а $\mathbf{x}(\infty)$ — выход.

Модели Хопфилда

14.4. Рассмотрим сеть Хопфилда, состоящую из пяти нейронов, в которой требуется сохранить следующие три ячейки фундаментальной памяти:

$$\boldsymbol{\xi}_1 = [+1, +1, +1, +1, +1]^T,$$
$$\boldsymbol{\xi}_2 = [+1, -1, -1, +1, -1]^T,$$
$$\boldsymbol{\xi}_3 = [-1, +1, -1, +1, +1]^T.$$

а) Вычислите матрицу синаптических весов размерностью 5×5.

б) С помощью асинхронной коррекции покажите, что все три ячейки фундаментальной памяти, $\boldsymbol{\xi}_1$, $\boldsymbol{\xi}_2$ и $\boldsymbol{\xi}_3$, удовлетворяют условию выравнивания.

в) Исследуйте производительность этой сети по извлечению данных, когда ей подается на вход зашумленная версия $\boldsymbol{\xi}_1$, в которой полярность второго элемента изменена.

14.5. Исследуйте возможность использования синхронной коррекции для измерения производительности извлечения информации в сети Хопфилда при условиях задачи 14.4.

14.6. а) Покажите, что

$$\begin{aligned}\boldsymbol{\xi}_1 &= [-1, -1, -1, -1, -1]^T,\\ \boldsymbol{\xi}_2 &= [-1, +1, +1, -1, +1]^T,\\ \boldsymbol{\xi}_3 &= [+1, -1, +1, -1, -1]^T\end{aligned}$$

также являются ячейками фундаментальной памяти сети Хопфилда, описанной в условиях задачи 14.4. Как эти ячейки связаны с ячейками, описанными в задаче 14.4?

б) Предположим, что первый элемент ячейки фундаментальной памяти $\boldsymbol{\xi}_3$ в задаче 14.4 замаскирован (т.е. принят равным нулю). Определите результирующий образ, который создаст модель Хопфилда. Сравните полученный результат с исходной формой $\boldsymbol{\xi}_3$.

14.7. Рассмотрим сеть Хопфилда, состоящую из двух нейронов. Матрица синаптических весов имеет следующий вид:

$$\mathbf{W} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.$$

Внешнее смещение, применяемое к нейронам, равно нулю. Четырьмя возможными состояниями сети могут быть

$$\begin{aligned}\mathbf{x}_1 &= [+1, +1]^T,\\ \mathbf{x}_2 &= [-1, +1]^T,\\ \mathbf{x}_3 &= [-1, -1]^T,\\ \mathbf{x}_4 &= [+1, -1]^T.\end{aligned}$$

а) Покажите, что состояния $\mathbf{x}_2$ и $\mathbf{x}_4$ являются устойчивыми, а состояния $\mathbf{x}_1$ и $\mathbf{x}_3$ представляют собой предельные циклы. При этом используйте следующий инструментарий: (1) условие выравнивания (устойчивости) и (2) функция энергии.

б) Какова длина предельного цикла, характеризующего состояния $\mathbf{x}_1$ и $\mathbf{x}_3$?

14.8. В этой задаче выведем формулу (14.55) емкости хранения практически без ошибок для модели Хопфилда, использующейся в качестве ассоциативной памяти.

а) Асимптотическое поведение функции ошибок приближенно описывается следующим выражением:

$$\operatorname{erf}(y) \simeq 1 - \frac{e^{-y^2}}{\sqrt{\pi}y} \quad \text{для больших } y.$$

Используя это приближение, покажите, что условная вероятность (14.53) может быть приближена следующим выражением:

$$P(v_j > 0 | \xi_{v,j} = +1) \simeq 1 - \frac{e^{-\rho/2}}{\sqrt{2\pi\rho}},$$

где ρ — отношение сигнал/шум. Покажите, что вероятность устойчивости состояния соответствующим образом аппроксимируется выражением

$$p_{\text{стаб.}} \simeq 1 - \frac{Ne^{-\rho/2}}{\sqrt{\pi\rho}}.$$

б) Второе слагаемое формулы $p_{\text{стаб.}}$, приведенной в части (а), является вероятностью того, что некоторый бит в ячейке фундаментальной памяти является неустойчивым. Для определения емкости хранения практически без ошибок недостаточно требовать только малости этого слагаемого. Оно должно быть мало по отношению к $1/N$, где N — размер сети Хопфилда. Покажите, что для этого отношение сигнал/шум должно удовлетворять условию

$$\rho > 2\log_e N + \frac{1}{2}\log_e(2\pi\rho).$$

в) Используя результат, полученный в части (б), покажите, что минимально допустимое значение отношения сигнал/шум для успешного извлечения большинства ячеек фундаментальной памяти составляет

$$\rho_{\min} = 2\log_e N.$$

Каково соответствующее значение $p_{\text{стаб.}}$?

г) Используя результат части (в), покажите, что

$$M_{\max} \simeq \frac{N}{2\log_e N},$$

что соответствует формуле (14.55).

д) Формула, выведенная в части (г) для емкости хранения, основана на допущении, что большая часть ячеек фундаментальной памяти является устойчивой. Для более строгого определения емкости хранения без ошибок будем требовать, чтобы *все* ячейки фундаментальной памяти извлекались без ошибок. Используя последнее определение, покажите, что максимальное число ячеек фундаментальной памяти, которое может храниться в сети Хопфилда, составляет [42]:

$$M_{\max} \simeq \frac{N}{4\log_e N}.$$

14.9. Покажите, что функция энергии сети Хопфилда может быть представлена в следующем виде

$$E = -\frac{N}{2}\sum_{v=1}^{M} m_v^2,$$

где m_v обозначает пересечения, определяемые как

$$m_v = \frac{1}{N}\sum_{j=1}^{N} x_j \xi_{v,j}, \quad v = 1, 2, \ldots, M,$$

где x_j — j-й элемент вектора состояния $\mathbf{x}$; $\xi_{v,j}$ — j-й элемент ячейки фундаментальной памяти $\boldsymbol{\xi}_v$; M — количество ячеек фундаментальной памяти.

14.10. Рассматриваемая сеть Хопфилда сконструирована для хранения двух ячеек фундаментальной памяти — $(+1, +1, -1, +1, +1)$ и $(+1, -1, +1, -1, +1)$.

Матрица синаптических весов этой сети имеет вид

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & -2 & 2 & 0 \\ 0 & -2 & 0 & -2 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{14.97}$$

а) Сумма собственных значений матрицы $\mathbf{W}$ равна нулю. Почему?

б) Пространство состояний этой сети является подпространством $\Re^5$. Определите конфигурацию этого подпространства.

в) Определите подпространство $\mathbf{M}$, натянутое на векторы фундаментальной памяти и нулевое подпространство $\mathbf{N}$ матрицы весов $\mathbf{W}$. Каковы устойчивые и ложные состояния этой сети?

(При решении задачи можно обратиться к работе, в которой содержится более подробное описание динамики этой сети [255].)

14.11. На рис. 14.26 показана кусочно-линейная форма немонотонной функции активации. Динамика восстановления в сети Хопфилда, использующей эту функцию активации, описывается следующим выражением:

$$\frac{d}{dt}\mathbf{v}(t) = -\mathbf{v}(t) + \mathbf{W}\mathbf{x}(t), \quad \mathbf{x}(t) = \operatorname{sgn}(\mathbf{v}(t)) - k\mathbf{v}(t),$$

где $\mathbf{v}(t)$ — вектор индуцированных локальных полей; $\mathbf{W}$ — матрица синаптических весов; $\mathbf{x}(t)$ — (выходной) вектор состояния; $-k$ — константа отрицательного наклона. Пусть $\bar{\mathbf{v}}$ — некоторое равновесное состояние сети, которое лежит в квадранте фундаментальной памяти $\boldsymbol{\xi}_1$, и пусть

$$\bar{\mathbf{x}} = \operatorname{sgn}(\bar{\mathbf{v}}) - k\bar{\mathbf{v}}.$$

Покажите, что $\bar{\mathbf{x}}$ характеризуется следующими тремя условиями:

а) $\sum_{i=1}^{N} \bar{x}_i \xi_{\mu,i} = 0, \quad \mu = 2, 3, \ldots, M,$

б) $\sum_{i=1}^{N} \bar{x}_i \xi_{1,i} = M,$

в) $\bar{x}_i < 1, \; i = 1, 2, \ldots, N,$

где $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_M$ — ячейки фундаментальной памяти, сохраненные в сети; $\xi_{\mu,i}$ — i-й элемент вектора $\boldsymbol{\xi}_\mu$; $\bar{x}_i$ — i-й элемент вектора $\bar{\mathbf{x}}$; N — количество нейронов.

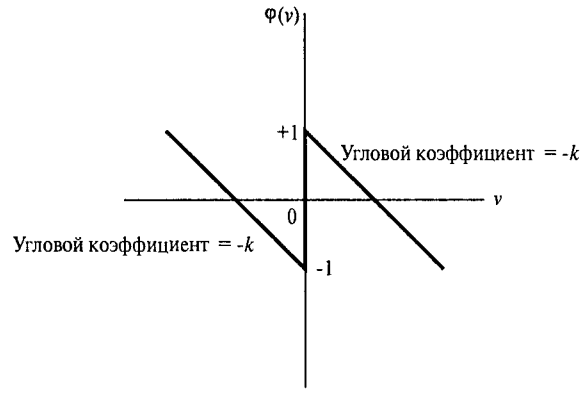


Рис. 14.26. Кусочно-линейная форма немонотонной функции активации

14.12. Рассмотрим простую нейродинамическую модель, описываемую системой уравнений

$$\frac{dv_j}{dt} = -v_j + \sum_i w_{ji}\varphi(v_i) + I_j, \quad j = 1, 2, \ldots, N.$$

Описанная система всегда сходится к единственному точечному аттрактору, при условии, что синаптические веса удовлетворяют соотношению

$$\sum_j \sum_i w_{ji}^2 < \frac{1}{(\max|\varphi'|)^2},$$

где $\varphi' = d\varphi/\mathrm{d}v_j$. Исследуйте правильность этого утверждения (для справки можно обратиться к работе, в которой выведено это условие [81]).

Теорема Коэна–Гроссберга

14.13. Рассмотрим функцию Ляпунова E, определяемую формулой (14.57). Покажите, что

$$\frac{dE}{dt} \leq 0$$

при условии выполнения соотношений (14.59)–(14.61).

14.14. В разделе 14.10 выведена функция Ляпунова модели BSB на основе применения теоремы Коэна–Гроссберга. В этом выводе опущены отдельные детали, приводящие к (14.73). Восстановите эти детали.

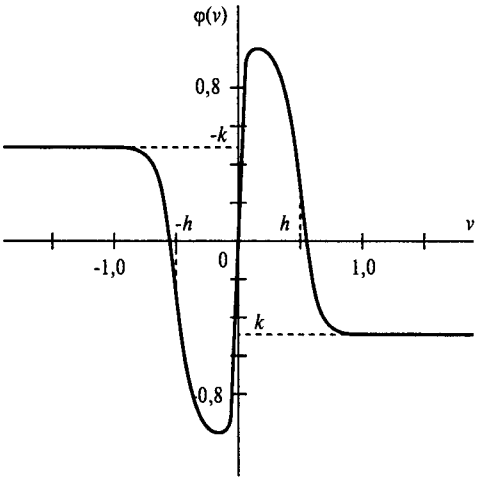


Рис. 14.27. График немонотонной функции активации

14.15. На рис. 14.27 показан график немонотонной функции активации, рассмотренной в [755]. Эта функция активации используется при построении сети Хопфилда вместо функции гиперболического тангенса. Применима ли теорема Коэна–Гроссберга к таким образом сконструированной ассоциативной памяти? Обоснуйте свой ответ.

ГЛАВА 15

Динамически управляемые рекуррентные сети

15.1. Введение

Как уже говорилось в предыдущей главе, *рекуррентными* (recurrent) называются нейронные сети, имеющие одну или несколько обратных связей. Обратные связи могут быть *локального* и *глобального* типов. В этой главе будет продолжено изучение рекуррентных сетей с глобальными обратными связями.

Если в качестве основного строительного блока используется многослойный персептрон, то применение обратной связи может принимать несколько форм. Во-первых, можно замкнуть выходной слой многослойного персептрона на его входной слой. Во-вторых, можно замкнуть выход скрытого слоя на вход. Так как многослойный персептрон может содержать несколько скрытых слоев, то последняя форма обратной связи может быть также сконфигурирована разными способами. Все это приводит к тому, что рекуррентные сети имеют богатый спектр архитектурных форм.

В основном рекуррентные сети используются в двух качествах.

- *Ассоциативная память* (associative memories).
- *Сети отображения вход-выход* (input-output mapping network).

Использование рекуррентных сетей в качестве ассоциативной памяти рассматривалось в главе 14; сейчас же речь пойдет об изучении их в качестве отображений вход-выход. Каким бы ни было использование рекуррентных сетей, в первую очередь подлежит рассмотрению вопрос их *устойчивости* (stability) (частично этот вопрос рассматривался в главе 14).

По определению входное пространство сетей отображается в выходное пространство. Для такого типа приложений рекуррентная сеть (temporally) отвечает во времени на применяемый извне входной сигнал. Таким образом, о сетях, рассматриваемых в настоящей главе, можно говорить как о *динамически управляемых рекуррентных сетях* (dynamically driven recurrent network). Более того, применение обратных связей позволяет использовать описание рекуррентных сетей в виде множества состояний,

что делает их удобными устройствами в таких областях, как нелинейное прогнозирование и моделирование, адаптивное выравнивание каналов связи, обработка речевых сигналов, управление предприятием и диагностика автомобильных двигателей. Как таковые рекуррентные сети являются альтернативой динамически управляемым сетям прямого распространения (см. главу 13). Из-за наличия дополнительного эффекта глобальной обратной связи они могут гораздо лучше проявлять себя в традиционных областях применения обычных нейронных сетей прямого распространения. Использование глобальной обратной связи позволяет значительно ослабить требования к объему используемой памяти.

Структура главы

Эта глава формально может быть разбита на четыре части: архитектура, теория, алгоритмы обучения и применение. В первой части (раздел 15.2) рассматриваются архитектуры рекуррентных сетей.

Во второй части (разделы 15.3–15.5) изложены теоретические аспекты рекуррентных сетей. В разделе 15.3 рассматривается модель в пространстве состояний и связанные с ней вопросы управляемости и наблюдаемости. В разделе 15.4 описывается модель нелинейной авторегрессии с внешними (exogenous) входами — эквивалент модели в пространстве состояний. В разделе 15.5 рассматриваются теоретические вопросы, связанные с вычислительной мощностью рекуррентных сетей.

Третья часть (разделы 15.6–15.12) посвящена алгоритмам обучения и связанным с ними вопросам. Она начинается с обзора рассматриваемой предметной области (раздел 15.6). Далее, в разделе 15.7, детально описывается обратное распространение во времени, которое основано на материале, представленном в главе 4. В разделе 15.8 представлен еще один популярный алгоритм: рекуррентное обучение в реальном времени. В разделе 15.9 приведен краткий обзор классической теории фильтров Калмана, а в разделе 15.10 описывается алгоритм несвязной расширенной фильтрации Калмана. Компьютерное моделирование последнего алгоритма, применяемого для рекуррентного обучения, будет выполнено в разделе 15.11. Градиентное рекуррентное обучение порождает проблему обращающихся в нуль градиентов, которая рассматривается в разделе 15.12.

Четвертая, заключительная, часть настоящей главы (разделы 15.13 и 15.14) посвящена важнейшим областям применения рекуррентных сетей. В разделе 15.13 описывается задача идентификации систем, а в разделе 15.14 — адаптивное управление на основе модели.

Глава завершается обобщающими выводами и рассуждениями.

15.2. Архитектуры рекуррентных сетей

Как уже говорилось во введении, архитектурное строение рекуррентных сетей может принимать множество различных форм. В этом разделе описываются четыре основные архитектуры таких сетей, каждая из которых соответствует некоторой частной форме глобальной обратной связи[1]. Эти архитектуры имеют следующие общие характеристики.

- Все они состоят из *статического* многослойного персептрона или его составных частей.
- Все они используют способность многослойного персептрона выступать в качестве отображения вход-выход (нелинейного оператора).

Рекуррентная модель "вход-выход"

На рис. 15.1 показана архитектура обобщенной рекуррентной сети, построенной на базе многослойного персептрона. Эта модель имеет единственный вход, который применяется к памяти на линиях задержки, состоящей из q элементов. Она имеет единственный выход, замкнутый на вход через память на линиях задержки, которая также состоит из q элементов. Содержимое этих двух блоков памяти используется для питания входного слоя персептрона. Вход модели обозначается как $u(n)$, а соответствующий выход — $y(n+1)$. Это значит, что выход модели упреждает ее вход на одну единицу времени. Таким образом, вектор сигнала, подаваемый на вход персептрона, состоит из окна данных, состоящего из следующих элементов.

- Текущее и предыдущее значения входного сигнала: $u(n), u(n-1), \ldots, u(n-q+1)$, которые представляют сети, имеющие внешнее происхождение.
- Значения выходного сигнала $y(n), y(n-1), \ldots, y(n-q+1)$ в предшествующие моменты времени, от которых зависит выход модели $y(n+1)$.

Таким образом, рекуррентную сеть, показанную на рис. 15.1, можно рассматривать как *модель нелинейной авторегрессии с внешними входами* (nonlinear autoregressive with exogenous inputs model — NARX)[2]. Динамика модели NARX описывается следующим образом:

---

[1] Другие архитектуры рекуррентных сетей описаны в [85], [306], [522], [893].

[2] Модель NARX охватывает важный класс дискретных нелиниейных систем [634]. В контексте нейронных сетей он рассмотрен в [184], [645], [774], [989].

Было продемонстрировано, что модель NARX хорошо подходит для моделироваиия нелинейных систем, такнх как теплообмениики [184], водоочистные системы [1024], [1025], системы каталитического реформинга [1025], нелинейные колебательные процессы, связанные с передвижением многоногих роботов (multilegged locomotion) [1091], и грамматический вывод (grammatical inference) [355].

Модель NARX также называют нелинейной моделью авторегрессии со скользящим средним (nonlinear autoregressive-moving average model — NARMA), в которой термин "скользящее среднее" относится к входному сигналу.

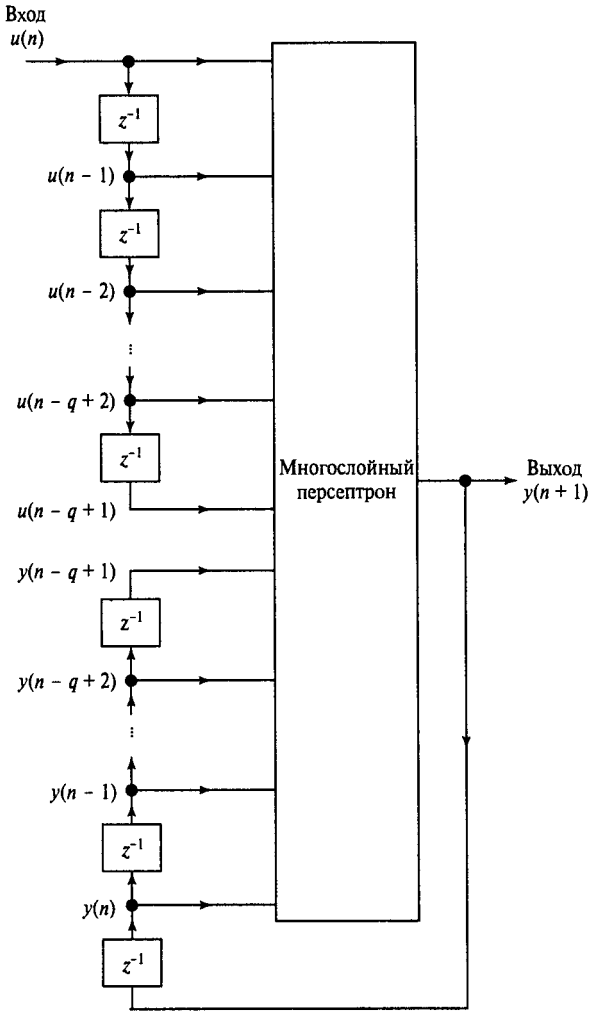


Рис. 15.1. Модель нелинейной регрессии с внешними входами (NARX)

$$y(n+1) = F(y(n), \ldots, y(n-q+1), u(n), \ldots, u(n-q+1)), \tag{15.1}$$

где F — некоторая нелинейная функция своих аргументов. Обратите внимание, на рис. 15.1 предполагается, что обе памяти на дискретной линии задержки имеют размер q. В общем случае эти размеры могут отличаться. Модель NARX детально описывается в разделе 15.4.

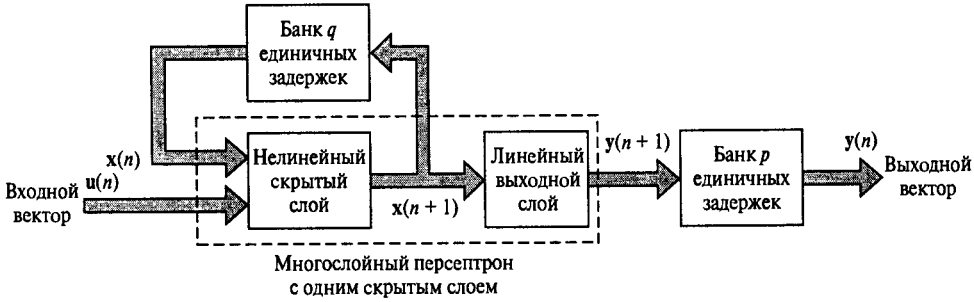


Рис. 15.2. Модель в пространстве состояний

Модель в пространстве состояний

На рис. 15.2 показана блочная диаграмма еще одной обобщенной рекуррентной сети, которая называется *моделью в пространстве состояний* (state-space model). Скрытые нейроны определяют *состояние* сети. Выход скрытого слоя замкнут на входной слой через банк единичных задержек. Входной слой сети состоит из объединения узлов обратной связи и узлов источника. Связь сети с внешней средой осуществляется через узлы источника. Количество единичных задержек, используемых для замыкания выхода скрытого слоя на входной слой, определяет *порядок модели* (order of model). Обозначим символом $\mathbf{u}(n)$ вектор входных сигналов размерности $m \times 1$ в момент времени n, а символом $\mathbf{x}(n)$ — вектор выходных сигналов скрытого слоя размерности $q \times 1$ в тот же момент времени. Тогда динамику этой модели можно описать следующей системой уравнений:

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{x}(n), \mathbf{u}(n)), \tag{15.2}$$

$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n), \tag{15.3}$$

где $\mathbf{f}(\cdot, \cdot)$ — некоторая нелинейная функция, характеризующая скрытый слой; $\mathbf{C}$ — матрица синаптических весов, характеризующих выходной слой. Скрытый слой этой сети нелинеен, а выходной — линеен.

Рекуррентная сеть на рис. 15.2 включает ряд рекуррентных архитектур в качестве частных случаев. Для примера рассмотрим *простую рекуррентную сеть* (simple recurrent network — SRN), описанную в работе Элмана [281] и показанную на рис. 15.3. Сеть Элмана имеет архитектуру, сходную с показанной на рис. 15.2, за исключением того факта, что выходной слой может быть нелинейным, а банк единичных задержек на выходе сети отсутствует.

Сеть Элмана содержит рекуррентные связи скрытых нейронов со слоем *контекстных элементов* (context unit), состоящим из единичных задержек. Эти контекстные элементы сохраняют выходы скрытых нейронов на один шаг времени, после чего

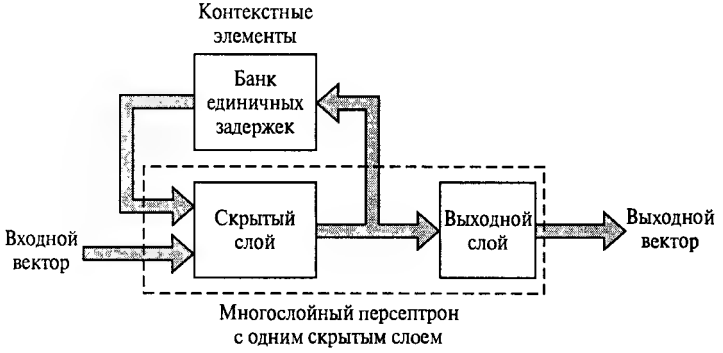


Рис. 15.3. Простая рекуррентная сеть SRN

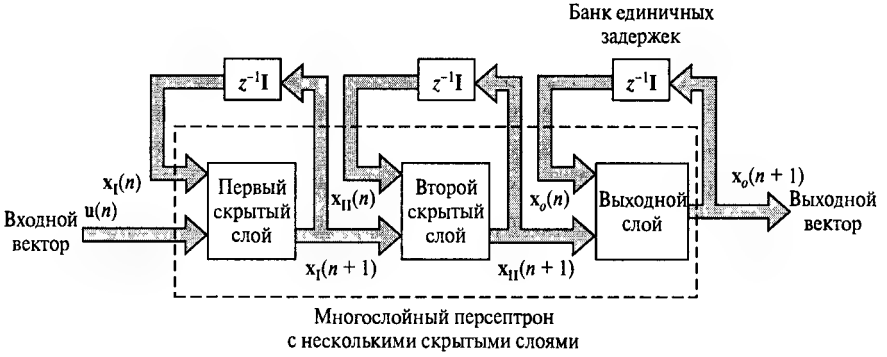


Рис. 15.4. Рекуррентный многослойный персептрон

передают их на входной слой. Таким образом, скрытые нейроны записывают свои предыдущие действия, что позволяет реализовать задачи обучения, разворачивающиеся во времени. Скрытые нейроны также передают информацию выходным нейронам, которые формируют реакцию сети на примененное извне возмущение. Так как данная природа обратной связи связана исключительно со скрытыми нейронами, они могут распространять повторные циклы информации по сети на протяжении множества шагов времени и таким образом открывать доступ к абстрактному представлению времени. Следовательно, простая рекуррентная сеть является чем-то бо́льшим, нежели записью на магнитную ленту предыдущих данных.

В [281] рассматривалось также использование простой рекуррентной сети (см. рис. 15.3) для поиска границ слов в последовательных потоках фонем без наличия каких-либо встроенных *ограничений представления* (representational constraint). На вход сети подавалась текущая фонема. Выход представлял собой наилучшее решение сети относительно того, какой будет следующая фонема в последовательности. Роль контекстных элементов обеспечивала *динамическая память* (dynamical memory), которая кодировала информацию, содержащуюся в последовательности фонем, что равноценно прогнозированию.

Рекуррентный многослойный персептрон

Третья рассматриваемая в настоящей главе рекуррентная архитектура известна как *рекуррентный многослойный персептрон* (recurrent multilayer perceptron — RMLP) [863]. Этот персептрон имеет один или несколько скрытых слоев (использование нескольких скрытых слоев определяется тем, что статичный многослойный персептрон обычно является более эффективным и компактным, чем использующий только один скрытый слой). Каждый вычислительный слой RMLP замкнут на себя собственной обратной связью. На рис. 15.4 показан частный случай RMLP с двумя скрытыми слоями[3].

Пусть вектор $\mathbf{x}_{\mathrm{I}}(n)$ обозначает выход первого скрытого слоя, вектор $\mathbf{x}_{\mathrm{II}}(n)$ — выход второго скрытого слоя и т.д. Обозначим символом $\mathbf{x}_o(n)$ выход выходного слоя. Тогда динамика RMLP в ответ на подачу на вход вектора $\mathbf{u}(n)$ в общем случае будет описываться следующей системой уравнений:

$$\begin{aligned}
\mathbf{x}_{\mathrm{I}}(n+1) &= \boldsymbol{\varphi}_{\mathrm{I}}(\mathbf{x}_{\mathrm{I}}(n), \mathbf{u}(n)), \\
\mathbf{x}_{\mathrm{II}}(n+1) &= \boldsymbol{\varphi}_{\mathrm{II}}(\mathbf{x}_{\mathrm{II}}(n), \mathbf{x}_{\mathrm{I}}(n+1)), \\
&\ldots \\
\mathbf{x}_o(n+1) &= \boldsymbol{\varphi}_o(\mathbf{x}_o(n), \mathbf{x}_K(n+1)),
\end{aligned} \tag{15.4}$$

где $\boldsymbol{\varphi}_{\mathrm{I}}(\cdot,\cdot)$, $\boldsymbol{\varphi}_{\mathrm{II}}(\cdot,\cdot)$, ..., $\boldsymbol{\varphi}_o(\cdot,\cdot)$ — функции активации, характеризующие первый, второй и другие скрытые слои, а также выходной слой RMLP; K — общее количество скрытых слоев в сети.

Описанный таким образом RMLP является обобщением сети Элмана, показанной на рис. 15.3, и модели в пространстве состояний, показанной на рис. 15.2, так как ни выходной слой, ни какой-либо из скрытых слоев не ограничен некоторой конкретной формой функции активации.

Сеть второго порядка

При описании модели в пространстве состояний (см. рис. 15.2) для обозначения количества скрытых нейронов, выходы которых замкнуты на выходной слой через банк единичных задержек, использовался термин "порядок".

В другом контексте термин "порядок" иногда используется для обозначения способа определения индуцированного локального поля нейрона. Для примера рассмотрим многослойный персептрон, индуцированное локальное поле v_k нейрона k в котором определяется следующим выражением:

---

[3] Рекуррентный многослойный персептрон, показанный на рис. 15.4, является обобщением рекуррентной сети, описанной в [522].

$$v_k = \sum_j w_{a,kj} x_j + \sum_i w_{b,ki} u_i, \tag{15.5}$$

где x_j — сигнал обратной связи, направленный от скрытого нейрона j; u_i — сигнал источника, применяемый к нейрону i входного слоя. Символами w обозначаются соответствующие синаптические веса сети. Нейрон, показанный на рис. 15.5, называют *нейроном первого порядка* (first-order neuron). Когда же индуцированное локальное поле формируется с использованием перемножения:

$$v_k = \sum_i \sum_j w_{kij} x_i u_j, \tag{15.6}$$

то такой нейрон называют *нейроном второго порядка*. Нейрон второго порядка k использует один вес w_{kji}, который связывает его с входными узлами i и j.

Нейроны второго порядка лежат в основе *рекуррентной сети второго порядка* [357], пример которой показан на рис. 15.5. Эта сеть принимает упорядоченную по времени последовательность входных сигналов и имеет динамику, описываемую следующей системой уравнений:

$$v_k(n) = b_k + \sum_i \sum_j w_{kij} x_i(n) u_j(n), \tag{15.7}$$

$$x_k(n+1) = \varphi(v_k(n)) = \frac{1}{1 + \exp(-v_k(n))}, \tag{15.8}$$

где $v_k(n)$ — индуцированное локальное поле нейрона k; b_k — соответствующее внешнее смещение (bias); $x_k(n)$ — состояние (выход) нейрона k; $u_j(n)$ — входной сигнал, поступающий на узел источника j; w_{kij} — вес нейрона второго порядка k.

Уникальным свойством рекуррентной сети второго порядка (см. рис. 15.5) является то, что произведение $x_j(n)u_j(n)$ представляет собой пару {состояние, вход}; положительный вес w_{kij} описывает наличие *перехода состояния* (state transition) {состояние, вход}→{следующее состояние}, а отрицательное значение веса — отсутствие такого перехода. Переход состояния описывается выражением

$$\delta(x_i, u_j) = x_k. \tag{15.9}$$

В свете этого соотношения сети второго порядка часто используются для представления и обучения *детерминированных конечных автоматов*[4] (deterministic finite-state automata — DFA). Автомат DFA — это устройство обработки информации с конечным

[4] В [801] показано, что с помощью рекуррентной сети второго порядка любой конечный автомат может быть отображен в такую сеть, и при этом гарантируется корректная классификация временных последовательностей конечной длины.

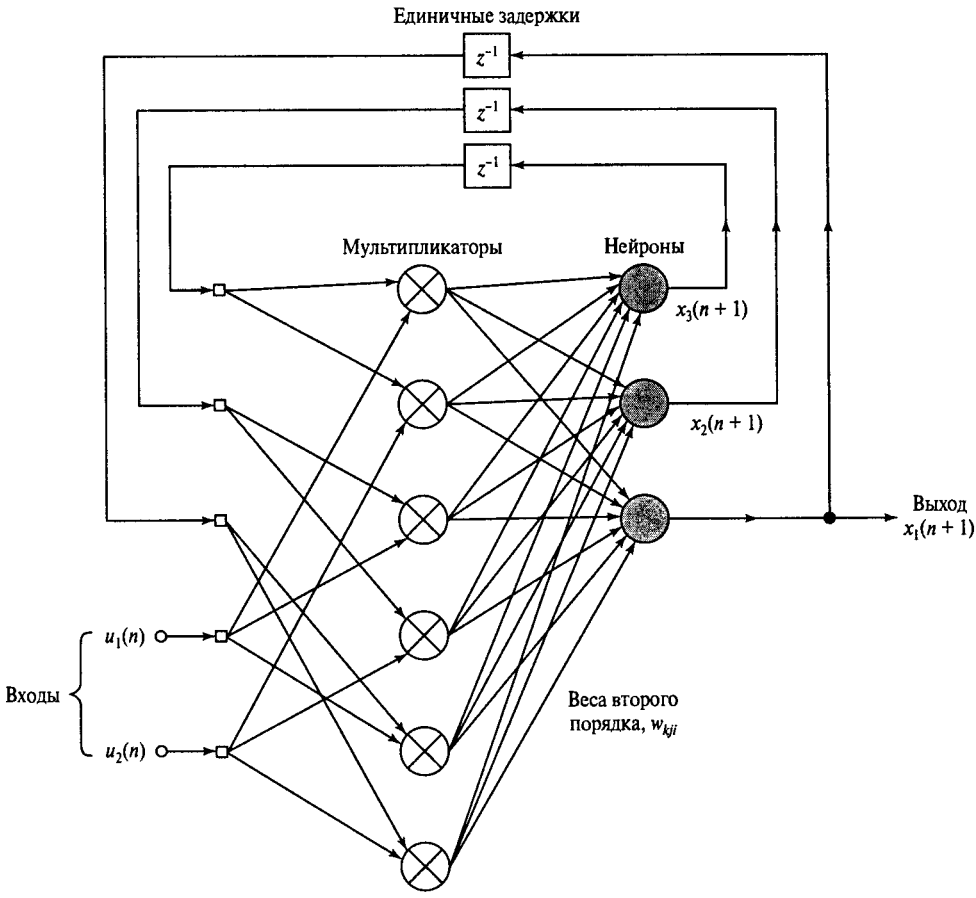


Рис. 15.5. Рекуррентная сеть второго порядка. Bias-нейроны исключены из рассмотрения для упрощения представления. Сеть содержит 2 входных нейрона, 3 нейрона состояния и, следовательно, требует $2 \times 3 = 6$ мультипликаторов

количеством состояний. Более подробная информация о связи между нейронными сетями и автоматами содержится в разделе 15.5.

Архитектуры рекуррентных сетей, описанные в этом разделе, подразумевают наличие глобальной обратной связи. Как уже говорилось во введении, архитектуры рекуррентных сетей могут быть построены и на использовании только локальных обратных связей. Свойства этого класса сетей хорошо описаны в [1060] (см. также задачу 15.7).

15.3. Модель в пространстве состояний

Понятие *пространства* играет жизненно важную роль в математическом описании динамических систем. Состояние динамической системы формально определяется как *множество величин, которые объединяют в себе всю информацию о прошлом системы, которая необходима для однозначного описания ее будущего, за исключением чисто внешних эффектов, возникающих при применении возбуждения*. Пусть вектор $\mathbf{x}(n)$ размерности $q \times 1$ обозначает состояние нелинейной системы дискретного времени; вектор $\mathbf{u}(n)$ размерности $m \times 1$ обозначает входной сигнал, применяемый к системе, а вектор $\mathbf{y}(n)$ размерности $p \times 1$ — соответствующий выход системы. В математических терминах динамика такой системы (в предположении отсутствия шума) описывается следующей системой нелинейных уравнений [1006]:

$$\mathbf{x}(n+1) = \boldsymbol{\varphi}(\mathbf{W}_a\mathbf{x}(n), \mathbf{W}_b\mathbf{u}(n)), \tag{15.10}$$

$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n), \tag{15.11}$$

где $\mathbf{W}_a$ — матрица размерности $q \times q$; $\mathbf{W}_b$ — матрица размерности $q \times (m+1)$; $\mathbf{C}$ — матрица размерности $p \times q$; φ: $\Re^q \rightarrow \Re^q$ — диагональное отображение вида

$$\varphi : \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_q \end{bmatrix} \rightarrow \begin{bmatrix} \varphi(x_1) \\ \varphi(x_2) \\ \cdots \\ \varphi(x_q) \end{bmatrix} \tag{15.12}$$

для покомпонентного описания некоторой нелинейности φ: $\Re \rightarrow \Re$. Пространства $\Re^m$, $\Re^p$ и $\Re^q$ называются соответственно *входным* (input), *выходным* (output) и *пространством состояний* (state space). Размерность пространства состояний q является *порядком системы* (order). Таким образом, модель в пространстве состояний, показанная на рис. 15.2, является *рекуррентной моделью порядка p с m входами и p выходами*. Уравнение (15.10) является *уравнением процесса* (process equation), описываемого этой моделью, а (15.11) — ее *уравнением измерения* (measurement equation). Уравнение процесса (15.10) является частным случаем уравнения (15.2).

Рекуррентная сеть, показанная на рис. 15.2, основана на использовании статического многослойного персептрона и двух блоков памяти с дискретной линией задержки. Она является одним из методов реализации нелинейной системы с обратной связью, описываемой системой (15.10)–(15.12). Обратите внимание, что на рис. 15.2 *за формирование состояния рекуррентной сети отвечают только те нейроны многослойного персептрона, выходы которых замкнуты на входной слой через блоки задержки*. Таким образом, из определения состояния исключаются выходные нейроны.

Можно предложить следующую интерпретацию матриц $\mathbf{W}_a$, $\mathbf{W}_b$ и $\mathbf{C}$, а также нелинейной функции $\varphi(\cdot)$.

- Матрица $\mathbf{W}_a$ содержит синаптические веса q нейронов скрытого слоя, которые соединены с узлами обратной связи входного слоя. Матрица $\mathbf{W}_b$ содержит синаптические веса q нейронов скрытого слоя, которые соединены с узлами источника входного слоя. Предполагается, что внешнее смещение, применяемое к скрытым нейронам, учтено в матрице весов $\mathbf{W}_b$.
- Матрица $\mathbf{C}$ содержит синаптические веса p линейных нейронов выходного слоя, которые соединяют их со скрытыми нейронами. Предполагается, что внешнее смещение, применяемое к выходным нейронам, учтено в матрице весов $\mathbf{C}$.
- Нелинейная функция $\varphi(\cdot)$ представляет сигмоидальные функции активации скрытых нейронов. Она обычно принимает форму функции гиперболического тангенса:

$$\varphi(x) = \tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{15.13}$$

или логистической функции:

$$\varphi(x) = \frac{1}{1 + e^{-x}}. \tag{15.14}$$

Важным свойством рекуррентных сетей, описываемых моделью в пространстве состояний (15.10) и (15.11), является то, что они могут *аппроксимировать* достаточно широкий класс нелинейных динамических систем. Однако эта аппроксимация достоверна только на компактных подмножествах пространства состояний и на конечных интервалах времени и не отражает некоторые интересные динамические характеристики [1006].

Пример 15.1

Для того чтобы проиллюстрировать композицию матриц $\mathbf{W}_a$, $\mathbf{W}_b$ и $\mathbf{C}$, рассмотрим *полносвязную рекуррентную сеть* (fully connected recurrent network), показанную на рис. 15.6. В ней контуры обратной связи начинаются в нейроиах скрытого слоя. В данном примере параметры m =2, q =3 и $p = 1$. Матрицы $\mathbf{W}_a$ и $\mathbf{W}_b$ определены следующим образом:

$$\mathbf{W}_a = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix},$$

$$\mathbf{W}_b = \begin{bmatrix} b_1 & w_{14} & w_{15} \\ b_2 & w_{24} & w_{25} \\ b_3 & w_{34} & w_{35} \end{bmatrix}.$$

Обратите внимание, что в первом столбце матрицы $\mathbf{W}_b$ сосредоточены внешние смещения b_1, b_2 и b_3, применяемые к нейронам 1, 2 и 3 соответственно. Матрица $\mathbf{C}$ в данном случае представляет собой вектор-строку

$$\mathbf{C} = [1, 0, 0].$$

■

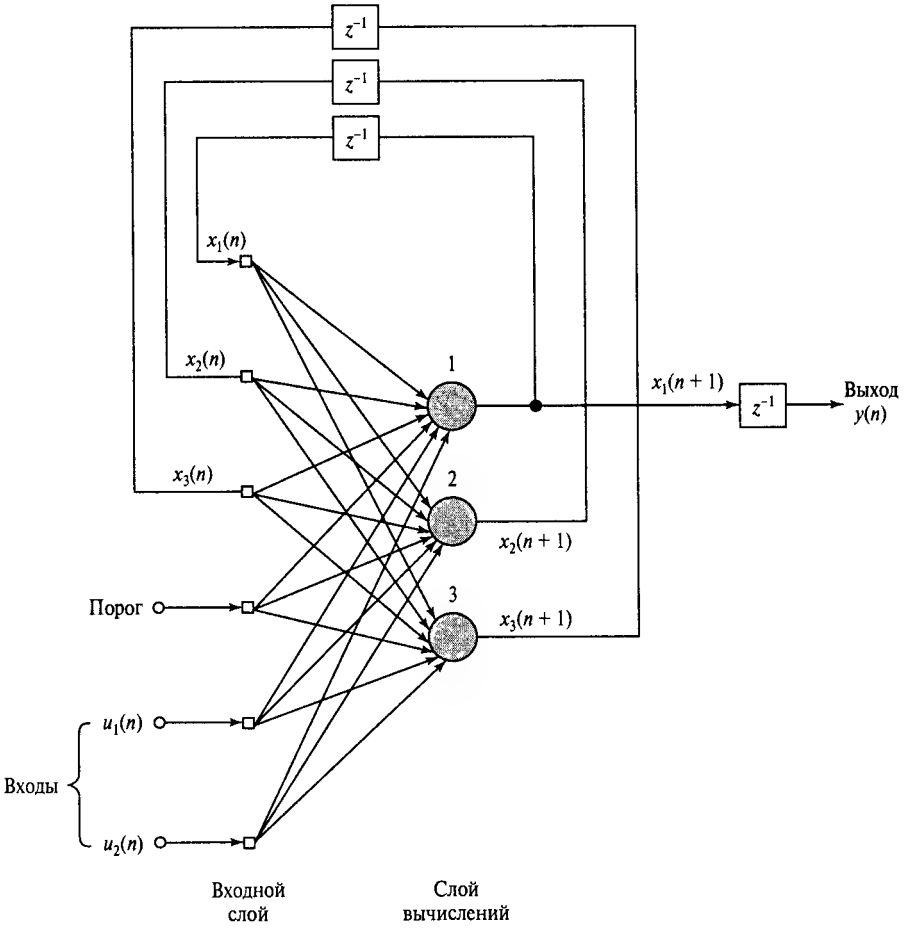


Рис. 15.6. Полносвязная рекуррентная сеть с двумя входами, двумя скрытыми нейронами и одним выходным нейроном

Управляемость и наблюдаемость

При изучении теории систем самыми важными рассматриваемыми вопросами являются устойчивость, управляемость и наблюдаемость. Устойчивость уже обсуждалась в предыдущей главе, поэтому не будем к ней больше возвращаться. В данном разделе рассмотрим вопросы управляемости и наблюдаемости, так как они обычно взаимосвязаны.

Как уже говорилось ранее, многие рекуррентные сети можно представить моделью в пространстве состояний (см. рис. 15.2), в которой состояние описывается выходом скрытого слоя, замкнутого обратной связью на входной слой через блок единичных задержек. В этом контексте важно знать, является ли сеть управляемой и наблюдаемой. Управляемость связана с вопросом возможности управления динами-

кой рекуррентной сети. Наблюдаемость связана с вопросом возможности наблюдения результата действия управления, примененного к сети. В этом смысле наблюдаемость является дуальной к управляемости.

Рекуррентная сеть называется *управляемой* (controllable), если из любого начального состояния путем управления можно привести систему в желаемое состояние за конечное число шагов. В данном определении выход системы не принимается в расчет. Рекуррентная сеть называется *наблюдаемой*, если состояние сети можно определить из конечного множества наблюдений входного и выходного сигналов. Более строгие определения управляемости и наблюдаемости выходят за рамки настоящей книги[5]. Здесь ограничимся только *локальными* формами управляемости и наблюдаемости (под локальностью понимается то, что эти понятия применяются только в окрестности равновесного состояния сети) [636].

Состояние $\bar{\mathbf{x}}$ называется *состоянием равновесия* (equilibrum state) системы (15.10), если для любого входа $\mathbf{u}$ выполняется условие

$$\bar{\mathbf{x}} = \boldsymbol{\varphi}(\mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}}). \tag{15.15}$$

Без потери общности можно положить $\bar{\mathbf{x}} = \mathbf{0}$ и $\bar{\mathbf{u}} = \mathbf{0}$. Тогда равновесное состояние будет описываться следующим образом:

$$\mathbf{0} = \boldsymbol{\varphi}(\mathbf{0}).$$

Другими словами, точка равновесия может быть представлена началом координат $(\mathbf{0}, \mathbf{0})$.

Без потери общности можно также упростить представление, ограничившись системой всего с *одним входом и одним выходом* (single input single output — SISO). Тогда равенства (15.10) и (15.11) можно переписать следующим образом:

$$\mathbf{x}(n+1) = \boldsymbol{\varphi}(\mathbf{W}_a\mathbf{x}(n), \mathbf{w}_b u(n)), \tag{15.16}$$

$$y(n) = \mathbf{c}^T\mathbf{x}(n), \tag{15.17}$$

где $\mathbf{w}_b$ и $\mathbf{c}$ являются векторами размерности $q \times 1$; $u(n)$ — скалярным входом; $y(n)$ — скалярным выходом. Поскольку $\boldsymbol{\varphi}$ является непрерывно дифференцируемой для случая использования сигмоидальных функций (15.13) и (15.14), ее можно *линеаризовать*, разложив в ряд Тейлора в окрестности точки равновесия $\bar{\mathbf{x}} = \mathbf{0}$ и $\bar{\mathbf{u}} = \mathbf{0}$ и ограничившись только слагаемыми первого порядка:

$$\delta\mathbf{x}(n+1) = \boldsymbol{\varphi}'(\mathbf{0})\mathbf{W}_a\delta\mathbf{x}(n) + \boldsymbol{\varphi}'(\mathbf{0})\mathbf{w}_b\delta u(n), \tag{15.18}$$

---

[5] Строгий подход к определению управляемости и наблюдаемости можно найти в [531], [638], [1008], [1178].

где $\delta\mathbf{x}(n)$ и $\delta u(n)$ являются малыми отклонениями состояния и входа соответственно, а матрица $\boldsymbol{\varphi}'(\mathbf{0})$ — Якобианом $\boldsymbol{\varphi}(\mathbf{v})$ по отношению к своему аргументу $\mathbf{v}$, вычисленным в точке $\mathbf{v} = \mathbf{0}$. Такую линеаризованную систему можно описать следующей системой уравнений:

$$\delta\mathbf{x}(n+1) = \mathbf{A}\delta\mathbf{x}(n) + \mathbf{b}\delta u(n)s, \tag{15.19}$$

$$\delta y(n) = \mathbf{c}^T\delta\mathbf{x}(n), \tag{15.20}$$

где матрица $\mathbf{A}$ размерности $q \times q$ и вектор $\mathbf{b}$ размерности $q \times 1$ определяются следующим образом:

$$\mathbf{A} = \boldsymbol{\varphi}'(\mathbf{0})\mathbf{W}_a, \tag{15.21}$$

$$\mathbf{b} = \boldsymbol{\varphi}'(\mathbf{0})\mathbf{w}_b. \tag{15.22}$$

Уравнения состояния (15.19) и (15.20) представлены в стандартной линейной форме. Следовательно, к ним можно применить хорошо известные результаты, относящиеся к управляемости и наблюдаемости динамических систем, полученные в стандартной части математической теории управления.

Локальная управляемость

Из линеаризованного уравнения (15.19) видно, что его последовательное применение приводит к системе следующих уравнений:

$$\begin{aligned}
\delta\mathbf{x}(n+1) &= \mathbf{A}\delta\mathbf{x}(n) + \mathbf{b}\delta u(n),\\
\delta\mathbf{x}(n+2) &= \mathbf{A}\delta\mathbf{x}(n+1) + \mathbf{b}\delta u(n+1),\\
&\ldots\\
\delta\mathbf{x}(n+q) &= \mathbf{A}^q\mathbf{b}\delta\mathbf{x}(n) + \mathbf{A}^{q-1}\mathbf{b}\delta\mathbf{x}(n+q-1) + \ldots + \mathbf{A}\mathbf{b}\delta u(n+1) + \mathbf{b}\delta u(n),
\end{aligned}$$

где q — размерность пространства состояний. Таким образом, можно утверждать следующее [636].

Линеаризованная система (15.19) является управляемой, если матрица

$$\mathbf{M}_c = [\mathbf{A}^{q-1}\mathbf{b}, \ldots, \mathbf{A}\mathbf{b}, \mathbf{b}] \tag{15.23}$$

имеет ранг q (т.е. полный ранг), так как тогда линеаризованное уравнение процесса (15.19) будет иметь единственное решение.

Матрица $\mathbf{M}_c$ называется *матрицей управляемости* (controllability matrix) линеаризованной системы.

Пусть рекуррентная сеть, описываемая уравнениями (15.16) и (15.17), управляется последовательностью входных сигналов $\mathbf{u}_q(n)$ вида

$$\mathbf{u}_q(n) = [u(n), u(n+1), \ldots, u(n+q-1)]^T. \tag{15.24}$$

Отсюда можно рассматривать отображение

$$\mathbf{G}(\mathbf{x}(n), \mathbf{u}_q(n)) = (\mathbf{x}(n), \mathbf{x}(n+q)), \tag{15.25}$$

где $\mathbf{G}$: $\Re^{2q} \rightarrow \Re^{2q}$. В задаче 15.4 показано, что

- состояние $\mathbf{x}(n+q)$ является *вложенной* (nested) нелинейной функцией своего прошлого значения $\mathbf{x}(n)$ и входов $u(n), u(n+1), \ldots, u(n+q-1)$;
- Якобиан $\mathbf{x}(n+q)$ по отношению к $\mathbf{u}_q(n)$, вычисленный в начале координат, равен матрице управляемости $\mathbf{M}_c$ из равенства (15.23).

Якобиан отображения $\mathbf{G}$ по отношению к $\mathbf{x}(n)$ и $\mathbf{u}_q(n)$, вычисленный в начале координат $(\mathbf{0}, \mathbf{0})$, можно выразить следующим образом:

$$\mathbf{J}_{(\mathbf{0},\mathbf{0})}^{(c)} = \begin{bmatrix} \left(\frac{\partial \mathbf{x}(n)}{\partial \mathbf{x}(n)}\right)_{(\mathbf{0},\mathbf{0})} & \left(\frac{\partial \mathbf{x}(n+q)}{\partial \mathbf{x}(n)}\right)_{(\mathbf{0},\mathbf{0})} \\ \left(\frac{\partial \mathbf{x}(n)}{\partial \mathbf{u}_q(n)}\right)_{(\mathbf{0},\mathbf{0})} & \left(\frac{\partial \mathbf{x}(n+q)}{\partial \mathbf{u}_q(n)}\right)_{(\mathbf{0},\mathbf{0})} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{X} \\ \mathbf{0} & \mathbf{M}_c \end{bmatrix}, \tag{15.26}$$

где $\mathbf{I}$ — единичная матрица; $\mathbf{0}$ — нулевая матрица, а матрица $\mathbf{X}$ не представляет интереса. Благодаря своей специфичной форме определитель этого Якобиана $\mathbf{J}_{(\mathbf{0},\mathbf{0})}^{(c)}$ равен произведению определителя единичной матрицы $\mathbf{I}$ и определителя матрицы управляемости $\mathbf{M}_c$. Если $\mathbf{M}_c$ имеет полный ранг, то таковой имеет и Якобиан $\mathbf{J}_{(\mathbf{0},\mathbf{0})}^{(c)}$.

Далее, приведем *теорему об обратной функции* (inverse function theorem), которая утверждает следующее [1095].

Рассмотрим отображение $\mathbf{f}$: $\Re^q \rightarrow \Re^q$ *и предположим, что каждый компонент* f *дифференцируем по своему аргументу в точке равновесия* $\mathbf{x}_0 \in \Re^q$ *и что* $\mathbf{y}_0 = \mathbf{f}(\mathbf{x}_0)$. *Тогда существует такое множество* $\mathbf{U} \in \Re^q$, *содержащее* $\mathbf{x}_0$, *и множество* $\mathbf{V} \in \Re^q$, *содержащее* $\mathbf{y}_0$, *что* $\mathbf{f}$ *будет представлять собой диффеоморфизм из* $\mathbf{U}$ *в* $\mathbf{V}$. *Если к тому же отображение* $\mathbf{f}$ *является гладким, то гладким будет и обратное отображение* $\mathbf{f}^{-1}$: $\Re^q \rightarrow \Re^q$ *и* $\mathbf{f}$ *будет являться гладким диффеоморфизмом.*

Отображение $\mathbf{f}$: $\mathbf{U} \rightarrow \mathbf{V}$ называют *диффеоморфизмом*, если оно удовлетворяет следующим трем условиям.

1. $\mathbf{f}(\mathbf{U}) = \mathbf{V}$.
2. Отображение $\mathbf{f}$: $\mathbf{U} \rightarrow \mathbf{V}$ является однозначным (т.е. обратимым).
3. Каждый компонент обратного отображения $\mathbf{f}^{-1}$:$\mathbf{V} \rightarrow \mathbf{U}$ является непрерывно дифференцируемым по своему аргументу.

Возвращаясь к вопросу управляемости, можно идентифицировать отображение $\mathbf{f}(\mathbf{U})=\mathbf{V}$ из теоремы об обратной функции как отображение, определенное формулой (15.25). Используя теорему об обратной функции, можно утверждать, что если матрица управляемости $\mathbf{M}_c$ имеет ранг q, то локально существует обратное отображение:

$$(\mathbf{x}(n), \mathbf{x}(n+q)) = \mathbf{G}^{-1}(\mathbf{x}(n), \mathbf{u}_q(n)). \tag{15.27}$$

Уравнение (15.27), в свою очередь, говорит о том, что существует такая последовательность входных сигналов $\{\mathbf{u}_q(n)\}$, которая может локально перевести сеть из состояния $\mathbf{x}(n)$ в состояние $\mathbf{x}(n+q)$ за q шагов. Следовательно, можно сформулировать следующую *теорему о локальной управляемости* (local controllability theorem) [636].

Пусть рекуррентная сеть определяется системой уравнений (15.16) и (15.17) и пусть ее линеаризованная версия в окрестности начала координат (т.е. равновесной точке) определяется системой (15.19) и (15.20). Если линеаризованная система является управляемой, то рекуррентная сеть является локально управляемой в окрестности начала координат.

Локальная наблюдаемость

Периодически используя линеаризованные уравнения (15.19) и (15.20), можно записать:

$$\delta y(n) = \mathbf{c}^T \delta\mathbf{x}(n),$$
$$\delta y(n+1) = \mathbf{c}^T \delta\mathbf{x}(n+1) = \mathbf{c}^T \mathbf{A}\delta\mathbf{x}(n) + \mathbf{c}^T \mathbf{b}\delta\mathbf{x}(n),$$

$$\ldots$$

$$\delta y(n+q-1) = \mathbf{c}^T \mathbf{A}^{q-1}\delta\mathbf{x}(n) + \mathbf{c}^T \mathbf{A}^{q-2}\mathbf{b}\delta u(n) + \ldots + \mathbf{c}^T \mathbf{A}\mathbf{b}\delta u(n+q-3) +$$

$$+ \mathbf{c}^T \mathbf{b}\delta u(n+q-2),$$

где q — размерность пространства состояний. Таким образом, можно утверждать следующее [636].

Линеаризованная система, описываемая уравнениями (15.19) и (15.20), является наблюдаемой, если матрица

$$\mathbf{M}_o = [\mathbf{c}, \mathbf{c}\mathbf{A}^T, \ldots, \mathbf{c}(\mathbf{A}^T)^{q-1}] \tag{15.28}$$

имеет ранг q (т.е. полный ранг).

Эта матрица $\mathbf{M}_o$ называется *матрицей наблюдаемости* (observability matrix) линеаризованной системы.

Пусть рекуррентная сеть, описываемая уравнениями (15.16) и (15.17), управляется последовательностью входных сигналов, заданных следующим образом:

$$\mathbf{u}_{q-1}(n) = [u(n), u(n+1), \ldots, u(n+q-2)]^T. \tag{15.29}$$

Соответственно пусть

$$\mathbf{y}_q(n) = [y(n), y(n+1), \ldots, y(n+q-1)]^T \tag{15.30}$$

обозначает вектор выходного сигнала, производимого начальным состоянием $\mathbf{x}(n)$ и последовательностью входов $\mathbf{u}_{q-1}(n)$. Тогда можно рассмотреть отображение

$$\mathbf{H}(\mathbf{u}_{q-1}(n), \mathbf{x}(n)) = (\mathbf{u}_{q-1}(n), \mathbf{y}_q(n)), \tag{15.31}$$

где $\mathbf{H}$: $\Re^{2q-1} \rightarrow \Re^{2q-1}$. В задаче 15.5 показано, что Якобиан $\mathbf{y}_q(n)$ по $\mathbf{x}(n)$, вычисленный в начале координат $(\mathbf{0}, \mathbf{0})$, равен матрице наблюдаемости $\mathbf{M}_o$ (15.28). Таким образом, Якобиан $\mathbf{H}$ по $\mathbf{u}_{q-1}(n)$ и $\mathbf{x}(n)$ в начале координат $(\mathbf{0}, \mathbf{0})$ можно выразить следующим образом:

$$\mathbf{J}^{(o)}_{(\mathbf{0},\mathbf{0})} = \begin{bmatrix} \left(\frac{\partial \mathbf{u}_{q-1}(n)}{\partial \mathbf{u}_{q-1}(n)}\right)_{(\mathbf{0},\mathbf{0})} & \left(\frac{\partial \mathbf{y}_q(n)}{\partial \mathbf{u}_{q-1}(n)}\right)_{(\mathbf{0},\mathbf{0})} \\ \left(\frac{\partial \mathbf{u}_{q-1}(n)}{\partial \mathbf{x}(n)}\right)_{(\mathbf{0},\mathbf{0})} & \left(\frac{\partial \mathbf{y}_q(n)}{\partial \mathbf{x}(n)}\right)_{(\mathbf{0},\mathbf{0})} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{X} \\ \mathbf{0} & \mathbf{M}_o \end{bmatrix}, \tag{15.32}$$

где матрица $\mathbf{X}$ опять не представляет интереса. Определитель этого Якобиана $\mathbf{J}^{(o)}_{(\mathbf{0},\mathbf{0})}$ равен произведению определителя единичной матрицы $\mathbf{I}$ и определителя матрицы наблюдаемости $\mathbf{M}_o$. Если $\mathbf{M}_o$ имеет полный ранг, то таковой имеет и Якобиан $\mathbf{J}^{(o)}_{(\mathbf{0},\mathbf{0})}$. Применяя теорему об обратной функции, можно утверждать, что если матрица наблюдаемости $\mathbf{M}_o$ линеаризованной системы имеет полный ранг, то локально существует некоторое обратное отображение, определяемое следующим образом:

$$(\mathbf{u}_{q-1}(n), \mathbf{x}(n)) = \mathbf{H}^{-1}(\mathbf{u}_{q-1}(n), \mathbf{x}(n)). \tag{15.33}$$

В результате это уравнение утверждает, что в локальной окрестности начала координат $\mathbf{x}(n)$ является некоторой нелинейной функцией от $\mathbf{u}_{q-1}(n)$ и $\mathbf{y}_q(n)$ и что эта функция является наблюдателем данной рекуррентной сети.

Таким образом, можно сформулировать следующую *теорему о локальной наблюдаемости* (local observability theorem) [636].

Пусть рекуррентная сеть определяется системой уравнений (15.16) и (15.17) и пусть ее линеаризованная версия в окрестности начала координат (т.е. равновесной точке) определяется системой (15.19) и (15.20). Если линеаризованная система является наблюдаемой, то рекуррентная сеть является локально наблюдаемой в окрестности начала координат.

Пример 15.2

Рассмотрим модель в пространстве состояний с матрицей $\mathbf{A} = a\mathbf{I}$, где a — скаляр; $\mathbf{I}$ — единичная матрица. Тогда матрица управляемости (15.23) сводится к

$$\mathbf{M}_c = a[\mathbf{b}, \ldots, \mathbf{b}, \mathbf{b}].$$

Ранг этой матрицы равен единице. Исходя из этого, линеаризованная система с такой матрицей $\mathbf{A}$ не является управляемой.

Подставляя выражение $\mathbf{A}$=$a\mathbf{I}$ в формулу (15.28), получим следующую матрицу наблюдаемости:

$$\mathbf{M}_o = a[\mathbf{c}, \mathbf{c}, \ldots, \mathbf{c}],$$

ранг которой также равен единице. Следовательно, эта линеаризованная система не является также и наблюдаемой. ■

15.4. Нелинейная автогрессия с внешней моделью входов

Рассмотрим рекуррентную сеть с одним входом и одним выходом, динамика которой описывается уравнениями состояния (15.16) и (15.17). Требуется модифицировать эту модель к модели типа "вход-выход", эквивалентно представляющей данную рекуррентную сеть.

С помощью уравнений (15.16) и (15.17) можно показать, что выход $y(n+q)$ можно выразить в терминах состояния $\mathbf{x}(n)$ и вектора входных сигналов $\mathbf{u}_q(n)$ следующим образом (см. задачу 15.8):

$$y(n+q) = \Phi(\mathbf{x}(n), \mathbf{u}_q(n)), \tag{15.34}$$

где q — размерность пространства состояний, а Φ: $\Re^{2q} \rightarrow \Re$. Предполагая, что данная рекуррентная сеть является наблюдаемой, можно применить теорему о локальной наблюдаемости и записать:

$$\mathbf{x}(n) = \mathbf{\Psi}(\mathbf{y}_q(n), \mathbf{u}_{q-1}(n)), \tag{15.35}$$

где $\mathbf{\Psi} : \Re^{2q-1} \rightarrow \Re^q$. Подставляя (15.35) в (15.34), получим:

$$y(n+q) = \Phi(\mathbf{\Psi}(\mathbf{y}_q(n), \mathbf{u}_{q-1}(n)), \mathbf{u}_q(n)) = F(\mathbf{y}_q(n), \mathbf{u}_q(n)), \tag{15.36}$$

где $\mathbf{u}_{q-1}(n)$ содержится в $\mathbf{u}_q(n)$ в качестве его первых $(q-1)$ элементов, а нелинейное отображение F: $\Re^{2q} \rightarrow \Re$ является композицией Φ и $\mathbf{\Psi}$. Используя определения $\mathbf{y}_q(n)$ и $\mathbf{u}_q(n)$ из формул (15.30) и (15.29), можно переписать выражение (15.36) в расширенном виде:

$$y(n+q) = F(y(n+q-1), \ldots, y(n), u(n+q-1), \ldots, u(n)).$$

Подставляя вместо n выражение $n-q+1$, можно эквивалентно записать [772]:

$$y(n+1) = F(y(n), \ldots, y(n-q+1), u(n), \ldots, u(n-q+1)). \qquad (15.37)$$

Переводя это выражение на язык слов, некоторое нелинейное отображение F: $\Re^{2q} \rightarrow \Re$ существует, посредством чего текущее значение выхода $y(n+1)$ однозначно определяется в терминах прошлых значений выхода $y(n)$,..., $y(n-q+1)$ и текущего и прошлых значений входного сигнала $u(n)$,..., $u(n-q+1)$. Чтобы это представление "вход-выход" было эквивалентно модели в пространстве состояний (15.16) и (15.17), рекуррентная сеть должна быть наблюдаемой. Практическое значение этой эквивалентности состоит в том, что модель NARX (см. рис. 15.1), в которой глобальная обратная связь ограничена только выходным нейроном, способна имитировать соответствующую полную рекуррентную модель в пространстве состояний (см. рис. 15.2) (предполагая, что $m = 1$ и $p = 1$) без отличий в динамике.

Пример 15.3

Вернемся к полносвязной рекуррентной сети, изображенной на рис. 15.6. В целях настоящей дискуссии предположим, что один из входов (например, $u_2(n)$) является нулевым. И теперь наша сеть сокращается до модели SISO (с одним входом и одним выходом). Тогда, предполагая, что сеть является локально наблюдаемой, можно заменить эту полносвязную рекуррентную сеть моделью NARX, показанной на рис. 15.7. Эта эквивалентность сохраняется несмотря на тот факт, что модель NARX имеет ограниченную обратную связь, которая исходит из выходного нейрона, в то время как полносвязная рекуррентная сеть, показанная на рис. 15.6, содержит глобальные обратные связи, окружающие многослойный персептрон и исходящие из всех трех (2-х скрытых и 1-го выходного) нейронов. ■

15.5. Вычислительная мощность рекуррентных сетей

Рекуррентные сети (примерами которых являются модель в пространстве состояний (см. рис. 15.2) и модель NARX (см. рис. 15.1)) имеют внутреннюю способность имитировать *конечные автоматы* (finite-state automata).*Автомат* (automata) представляет собой абстракцию устройств обработки информации, таких как компьютеры. На самом деле история развития нейронных сетей и автоматов имеет много общего[6].

[6] Первая работа по нейронным сетям и автоматам (действительно последовательным реализациям машин-автоматов) также является первой работой, посвященной конечным автоматам, искусственному интеллекту

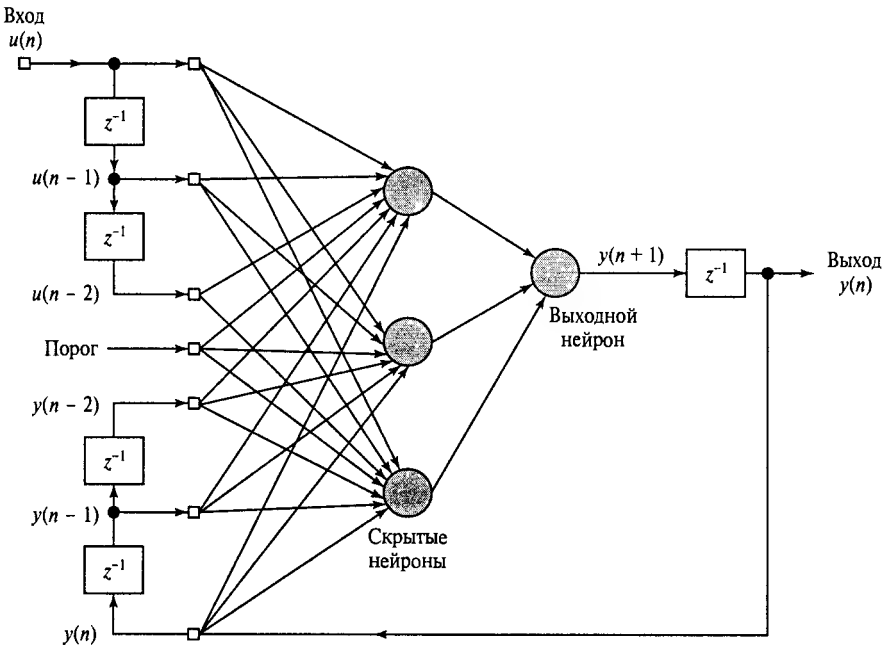


Рис. 15.7. Сеть NARX с $q = 3$ скрытыми нейронами

В [741] приводится следующее утверждение.

Любая машина с конечными состояниями эквивалента некоторой нейронной сети и может быть имитирована ею. Это значит, что для любой машины с конечными состояниями **M** *можно построить определенную нейронную сеть* $\mathbf{N}^{\mathbf{M}}$*, которая, если ее рассматривать как "черный ящик", будет работать в точности как машина* **M**.

Эта ранняя работа по рекуррентным сетям использовала для функции активации нейрона жесткую логику единичного скачка, а не мягкую сигмоидальную функцию.

Пожалуй, первая экспериментальная демонстрация того, сможет ли рекуррентная сеть обучиться особенностям, содержащимся в небольшой грамматике с конечны-

---

и рекуррентным нейронным сетям [714]. Рекуррентные сети (с немедленной обратной связью), описанные во второй части этой книги, в [562] были интерпретированы как конечные автоматы. Последний труд [972] появился под редакцией Шеннона и Маккарти (среди авторов этой удивительной книги были также Мур (Moore), Минский (Minsky), фон Нейман (von Neumann) и Аттли (Uttley)). Иногда [562] упоминается как первый труд, посвященный конечным машинам (finite-state machine) [827]. Автоматы и нейронные сети рассмотрены в [741].

Все ранние работы, посвященные автоматам и нейронным сетям, были связаны с синтезом, т.е. встраиванием автоматов в нейронные сети. Так как большая часть автоматов (реализованных как последовательные машины) требует обратной связи, нейронные сети также должны были быть рекуррентными. Заметим, что в ранних работах (за исключением [741]) не делалось четкого разграничения между автоматами (направленными, маркированными, ациклическими графами) и последовательными машинами (задержки логики и обратной связи) и больше внимания уделялось конечным автоматам. Проявлялся только небольшой интерес (за исключением [741]) к переходу к иерархии автоматов, развитию автоматов и машины Тьюринга.

После времен застоя в области нейронных сетей исследования автоматов возобновились в 1980-х годах. Эта работа может быть условно разбита на три направления: обучаемые автоматы; синтез автоматов, извлечение и структурирование знаний; представление. Впервые об автоматах и нейронных сетях упомииалось в [522].

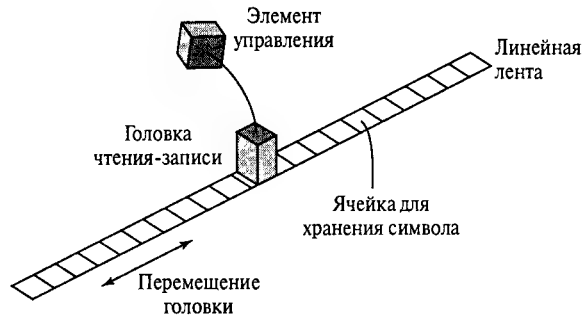


Рис. 15.8. Машина Тьюринга

ми состояниями, приведена в [198]. В данной работе простая рекуррентная сеть (см. рис. 15.3) была представлена строками, полученными из грамматики и требуемыми для прогнозирования на каждом шаге следующей буквы. Прогнозирование было контекстно-зависимым, так как каждая буква появлялась в грамматике дважды и каждый раз за ней следовали различные буквы. Было показано, что такая сеть способна выработать такое внутреннее представление в скрытых нейронах, которое соответствует состояниям автомата (машины с конечными состояниями). В [598] было представлено формальное доказательство того, что простая рекуррентная сеть имеет такую же вычислительную мощность, как и любая машина с конечными состояниями.

В обобщенном смысле вычислительная мощность рекуррентных сетей раскрывается двумя основными теоремами.

Теорема 1 [990].

Любую машину Тьюринга можно имитировать полносвязной рекуррентной сетью, построенной на нейронах с сигмоидальными функциями активации.

Машина Тьюринга (Turing machine) является абстрактным вычислительным устройством, изобретенным Тьюрингом в 1936 году. Она состоит из трех функциональных блоков (рис. 15.8): *элемента управления*, который может иметь одно из конечного числа возможных состояний; *линейной ленты* (в предположении, что она бесконечна в обоих направлениях), которая разбита на дискретные ячейки, а каждая ячейка способна хранить только один символ из конечного их множества; *головки чтения-записи*, которая перемещается вдоль ленты и считывает и записывает информацию в элемент управления [295]. В рамках нашей дискуссии будет достаточно сказать, что машина Тьюринга является абстракцией, которая имеет вычислительную мощность компьютера. Эта идея известна под названием *гипотезы Чарча–Тьюринга* (Church-Turing hypothesis).

Теорема 2 [989].

Сети NARX с одним слоем скрытых нейронов, имеющих ограниченную функцию активации с односторонним насыщением, и одним выходным нейроном могут имитировать полносвязные рекуррентные сети с ограниченными функциями активации с односторонним насыщением с учетом линейного замедления (linear slowdown).

Под "линейным замедлением" понимается следующее: если полносвязная рекуррентная сеть с N нейронами решает интересующую задачу за время T, то общее время, затрачиваемое эквивалентной сетью NARX, составит $(n+1)T$. Функция $\varphi(\cdot)$ называется *ограниченной функцией с односторонним насыщением* (bounded, one-sided saturated function — BOSS), если выполняются следующие условия.

1. Функция $\varphi(\cdot)$ имеет ограниченную область значений, т.е. для всех $x \in \Re$ выполняется $a \leq \varphi(x) \leq b$, где $a \neq b$.
2. Функция $\varphi(\cdot)$ имеет насыщение с левой стороны, т.е. существуют такие значения s и S, что $\varphi(x) = S$ для всех $x \leq s$.
3. Функция $\varphi(\cdot)$ не является константной, т.е. существуют такие x_1 и x_2, что $\varphi(x_1) \neq \varphi(x_2)$.

Пороговые (threshold) и кусочно-линейные функции удовлетворяют условиям BOSS. Однако в прямом смысле сигмоидальная функция не является функцией BOSS, так как нарушает условие 2. Тем не менее после небольшой модификации ее можно сделать таковой, записав (на примере логистической функции):

$$\varphi(x) = \begin{cases} \frac{1}{1+\exp(-x)}, & x > s, \\ 0, & x \leq s, \end{cases}$$

где $s \in \Re$. В результате при $x \leq s$ логистическая функция "обрезается".

Следствием из теорем 1 и 2 может служить следующее утверждение.

Сети NARX с одним скрытым слоем нейронов, имеющих BOSS-функции активации, и одним выходным нейроном эквивалентны сетям Тьюринга.

На рис. 15.9 графически изображены теоремы 1 и 2 и их следствие. Однако следует заметить, что если сетевая архитектура ограничена, то вычислительная мощность рекуррентной сети не является эквивалентной конечным автоматам [1012]. Ссылки на примеры ограниченных сетей представлены в примечании 7[7].

---

[7] Однослойные рекуррентные сети, использующие нейрон Мак-Каллока–Питца, не могли имнтировать какую-либо конечную машину [373]. В то же время простая рекуррентная сеть Элмана может это сделать [598]. Рекуррентные сети, содержащие только локальные обратные связи, не могут представлять *все* конечные машины [307], [352], [597].

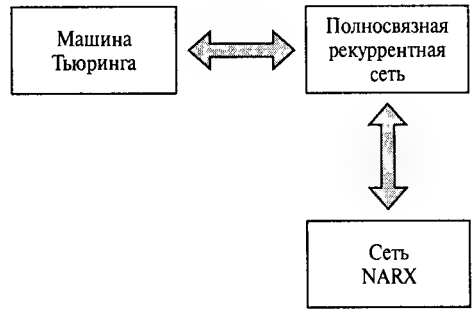


Рис. 15.9. Графическое изображение теорем 1 и 2 и их следствия

15.6. Алгоритмы обучения

Теперь обратимся к вопросу обучения рекуррентных сетей. В главе 4 уже говорилось, что существуют два режима обучения обычного (статического) многослойного персептрона: пакетный и последовательный. В пакетном режиме перед коррекцией свободных параметров вычисляется чувствительность сети для всего множества обучения. С другой стороны, в последовательном режиме настройка параметров выполняется после подачи каждого примера обучения. Подобно этому для обучения рекуррентной сети также существуют два метода обучения [1156].

Обучение по эпохам (epochwise training). В заданной эпохе рекуррентная сеть начинает свою работу с некоторого исходного состояния и развивается, пока не достигнет некоторого другого состояния. В этой точке обучение приостанавливается, и сеть сбрасывается в исходное состояние, после чего начинается следующая эпоха обучения. Исходное состояние не обязательно должно быть одинаковым для всех эпох. Важно только, чтобы исходное состояние каждой эпохи отличалось от конечного состояния предыдущей. Для примера рассмотрим использование рекуррентной сети для имитации работы *конечного автомата* (finite-state machine), т.е. устройства, число различных внутренних конфигураций (состояний) которого конечно. В такой ситуации будет вполне обоснованным использование обучения по эпохам, так как представляется удобная возможность имитации рекуррентной сетью множества различных начальных и конечных состояний автомата. При обучении рекуррентных сетей по эпохам термин "эпоха" используется в смысле, несколько отличном от того, который использовался для обычного многослойного персептрона. В текущей терминологии эпоха для рекуррентной сети соответствует одному примеру обучения для обычного многослойного персептрона.

Непрерывное обучение (continuous training). Второй метод обучения подходит для ситуаций, когда недоступны состояния сброса и (или) требуется обучение в реальном времени. Отличительной чертой непрерывного обучения является то, что сеть обучается во время самой реальной обработки сигнала, т.е. процесс обучения никогда не останавливается. Для примера рассмотрим использование рекуррентной сети для моделирования нестационарного процесса, такого как речевой сигнал. В такой

ситуации непрерывность работы сети не предполагает наличия удобного времени для останова обучения и последующего его начала с другими значениями свободных параметров сети.

Помня об этих двух режимах обучения, в последующих двух разделах рассмотрим следующие алгоритмы обучения рекуррентных сетей.

- Алгоритм обратного распространения во времени (back-propagation-through-time) рассматривается в разделе 15.7. Он работает в предположении того, что временные операции рекуррентной сети могут быть представлены посредством многослойного персептрона. Это может открыть путь для применения стандартного алгоритма обратного распространения. Обратное распространение во времени можно реализовать как по эпохам, так и в непрерывном режиме. Можно также комбинировать эти два режима.
- Алгоритм рекуррентного обучения в реальном времени, описанный в разделе 15.8, является производным от модели в пространстве состояний, описанной формулами (15.10) и (15.11).

Эти два алгоритма имеют ряд общих характеристик. Во-первых, оба они основаны на методе градиентного спуска, тогда как мгновенное значение функции стоимости (основанной на критерии среднеквадратической ошибки) минимизируется по синаптическим весам сети. Во-вторых, оба эти метода довольно просты в реализации, но могут медленно сходиться. В-третьих, они связаны тем, что представление алгоритма обратного распространения во времени в виде графа движения сигнала может быть получено *перестановкой* из графа движения сигнала определенной формы алгоритма рекуррентного обучения в реальном времени [109], [632].

(Непрерывное) обучение в реальном времени, основанное на градиентном спуске, использует *минимальный объем доступной информации*, а именно мгновенную оценку градиента функции стоимости по настраиваемому вектору параметров. Процесс обучения можно несколько ускорить, применив теорию фильтров Калмана, которая более эффективно использует информацию, содержащуюся в данных обучения. В разделе 15.10 описывается несвязный расширенный фильтр Калмана, посредством которого можно решать задачи динамического обучения, чересчур сложные для обычных методов градиентного спуска. Краткий обзор фильтров Калмана представлен в разделе 15.9. Обратите внимание, что несвязный расширенный фильтр Калмана применим как в статических сетях прямого распространения, так и в рекуррентных сетях.

Некоторые эвристики

Перед тем как приступить к описанию новых алгоритмов обучения, перечислим некоторые эвристики улучшения обучения рекуррентных сетей, которые используют методы градиентного спуска [351].

- Должен быть выдержан лексиграфический порядок примеров обучения, в котором самые короткие строки подаются сети в первую очередь.
- Обучение должно начинаться с небольшого примера, после чего их размер должен постепенно увеличиваться.
- Синаптические веса сети должны корректироваться только в том случае, если абсолютная ошибка при подаче текущего примера сети будет больше некоторого заранее заданного критерия.
- Рекомендуется использовать эвристику снижения (decay) весов (снижение весов — это одна из грубых форм регуляризации сложности, о которой говорилось в главе 4).

Первая эвристика представляет особый интерес. При ее реализации устраняется проблема обращения градиентов в нуль, которая возникает в рекуррентных сетях, обучаемых методами градиентного спуска. Об этой проблеме речь пойдет в разделе 15.12.

15.7. Обратное распространение во времени

Алгоритм обратного распространения во времени (back-propagation-through-time algorithm — BPTT), применяемый для обучения рекуррентных сетей, является расширением стандартного алгоритма обратного распространения[8]. Он может быть получен путем *развертывания* временных операций сети в многослойной сети прямого распространения, топология которой расширяется на один слой для каждого шага времени.

Для примера предположим, что **N** — рекуррентная сеть, требуемая для обучения временной задаче, которая начинается с момента времени n_0 и продолжается до момента времени n. Пусть **N*** — сеть прямого распространения, которая получается после развертывания временных операций рекуррентной сети **N**. Развернутая сеть **N*** связана с исходной сетью **N** следующим образом.

1. Для каждого шага времени из интервала $(n_0, n]$ сеть **N*** содержит слой, состоящий из K нейронов, где K — количество нейронов исходной сети **N**.
2. В каждом слое сети **N*** содержатся копии всех нейронов сети **N**.

[8] Идея, стоящая за алгоритмом обратного распространения во времени, заключается в том, что для любой рекуррентной сети можно создать сеть прямого распространения, обеспечивающую такую же динамику через некоторый интервал времени [745]. Алгоритм обратного распространения во времени впервые был описан в диссертационной работе Вербоса [1128] (см. также [1126]). Этот алгоритм независимо от Вербоса был сконструирован в [915]. Один из вариантов алгоритма обратного распространения во времени был описан в [1155]. Обзор этих алгоритмов и связанных с ними вопросов содержится в [1156].

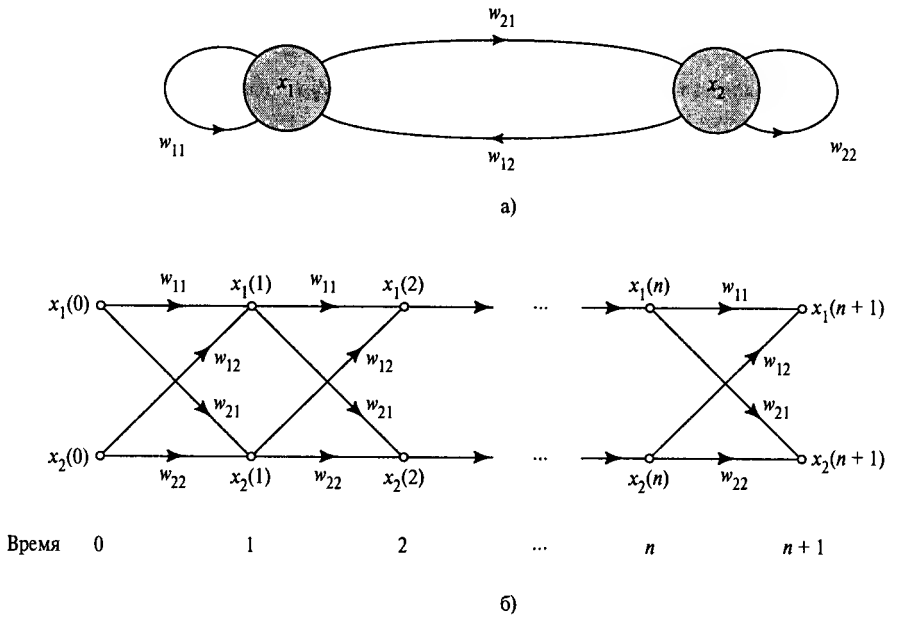


Рис. 15.10. Архитектурный граф рекуррентной сети **N**, состоящей из двух нейронов (а); граф передачи сигнала сети **N**, развернутый во времени (б)

3. Для каждого шага времени $l \in [n_0, n]$ синаптические связи от нейрона i слоя l к нейрону j слоя $l+1$ сети $\mathbf{N}^*$ являются копиями синаптических связей между нейронами i и j сети **N**.

Все эти моменты проиллюстрированы в следующем примере.

Пример 15.4

Рассмотрим рекуррентную сеть **N**, состоящую из двух нейронов (рис. 15.10, *а*). Для упрощения представления мы опустили операторы единичной задержки z^{-1}, которые следовало вставить в каждую из синаптических связей (в том числе и в собственные обратные связи) сети **N**. Пошагово раскрывая временные связи сети, получим граф передачи сигнала, который показан на рис. 15.10, *б* и в котором время начала операций равно $n_0 = 0$. Этот граф представляет многослойную сеть прямого распространения $\mathbf{N}^*$, в которой каждый новый слой добавляется на каждом шаге времени работы сети. ■

Применение процедуры развертывания приводит к двум совершенно различным реализациям обратного распространения во времени, в зависимости от того, какой режим обучения используется, по эпохам или непрерывный. Все эти методы рекуррентного обучения будут последовательно раскрыты далее в этом разделе.

Обратное распространение по эпохам во времени

Пусть множество данных, используемое для обучения рекуррентной сети, разбито на независимые эпохи, каждая из которых представляет интересующий отрезок времени. Пусть n_0 — время начала эпохи; n_1 — время ее окончания. Рассматривая такую эпоху, можно определить следующую функцию стоимости:

$$\mathbf{E}_{\text{общ.}}(n_0, n_1) = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{j \in \mathbf{A}} e_j^2(n), \tag{15.38}$$

где $\mathbf{A}$ — множество индексов j, относящееся к тем нейронам сети, для которых определены желаемые отклики; $e_j(n)$ — сигнал ошибки на выходе этих нейронов, измеренный по отношению к некоторому желаемому отклику. Требуется вычислить чувствительность этой сети, т.е. частные производные функции стоимости $\mathbf{E}_{\text{общ.}}(n_0, n_1)$ по синаптическим весам сети. Для этого можно использовать *алгоритм обратного распространения во времени по эпохам*, который построен на пакетном режиме стандартного обучения методом обратного распространения (см. главу 4). Алгоритм обучения по эпохам BPTT выполняется следующим образом [1155].

- Сначала осуществляется прямая передача данных по сети на интервале времени (n_0, n_1). Для записи входных данных состояние сети (т.е. ее синаптические веса) и желаемый отклик для этого интервала времени *сохраняются*.
- Для вычисления значений локальных градиентов выполняется единичная обратная передача через последнюю запись:

$$\delta_j(n) = -\frac{\partial \mathbf{E}_{\text{общ.}}(n_0, n_1)}{\partial v_j(n)} \tag{15.39}$$

для всех $j \in \mathbf{A}$ и $n_0 < n \leq n_1$. Это вычисление выполняется с помощью формулы

$$\delta_j(n) = \begin{cases} \varphi'(v_j(n)) e_j(n), & n = n_1, \\ \varphi'(v_j(n)) \left[e_j(n) + \sum_{k \in \mathbf{A}} w_{jk} \delta_k(n+1) \right], & n_0 < n < n_1, \end{cases} \tag{15.40}$$

где $\varphi'(\cdot)$ — производная функции активации по своему аргументу; $v_j(n)$ — индуцированное локальное поле нейрона j. Предполагается, что все нейроны сети имеют одну и ту же функцию активации $\varphi(\cdot)$. Вычисления по формуле (15.40) повторяются с момента времени n_1, шаг за шагом, пока не будет достигнут момент n_0. Количество выполняемых шагов равно количеству шагов времени в данной эпохе.

- После выполнения вычислений по методу обратного распространения до момента времени $n_0 + 1$ к синаптическим весам w_{ji} нейрона j применяется следующая коррекция:

$$\Delta w_{ji} = -\eta \frac{\partial \mathbf{E}_{\text{общ.}}(n_0, n_1)}{\partial w_{ji}} = \eta \sum_{n=n_0+1}^{n_1} \delta_j(n) x_i(n-1), \tag{15.41}$$

где η — параметр скорости обучения; $x_i(n-1)$ — входной сигнал, поданный на i-й синапс j-го нейрона в момент времени $n-1$.

Сравнивая описанную процедуру алгоритма BPTT по эпохам с пакетным режимом стандартного алгоритма обратного распространения, видим, что главное отличие заключается в том, что в первом случае желаемый отклик определяется для многих слоев сети, поскольку фактический выходной слой рекуррентной сети многократно воспроизводится при развертывании ее поведения во времени.

Усеченное обратное распространение во времени

Для использования алгоритма обратного распространения во времени в режиме реального времени в качестве минимизируемой функции стоимости используются мгновенные значения суммы квадратических ошибок:

$$\mathbf{E}(n) = \frac{1}{2} \sum_{j \in \mathbf{A}} e_j^2(n).$$

Как и в последовательном (стохастическом) режиме стандартного обучения методом обратного распространения, для вычисления соответствующей коррекции синаптических весов сети в каждый момент времени n используется градиент функции стоимости $\mathbf{E}(n)$, взятый с обратным знаком. Эти коррекции выполняются последовательно в ходе реальной работы сети. Однако, для того чтобы эта операция была выполнима с вычислительной точки зрения, сохраняется только история входных данных и состояний сети для фиксированного количества шагов времени, называемого *глубиной усечения* (truncation depth) и обозначаемого символом h. Любая информация, которая "старше" h шагов времени, считается неактуальной и может игнорироваться. Если не усекать вычисления, позволяя им накапливаться с момента времени начала процесса, то время вычислений и требуемый объем памяти возрастают линейно со временем и в конечном счете достигнут точки, в которой весь процесс обучения станет невыполнимым.

Эта вторая форма алгоритма называется *усеченным алгоритмом обратного распространения во времени* (BPTT(h)) [1155]. В нем локальный градиент нейрона j определяется следующим образом:

$$\delta_j(l) = -\frac{\partial \mathbf{E}(l)}{\partial v_j(l)} \text{ для всех } j \in A \text{ и } n-h < l \leq n, \tag{15.42}$$

что, в свою очередь, приводит к формуле

$$\delta_j(l) = \begin{cases} \varphi'(v_j(l))e_j(l), & l = n, \\ \varphi'(v_j(l)) \left[\sum\limits_{k \in \mathbf{A}} w_{kj}(l)\delta_k(l+1) \right], & n-h < l < n. \end{cases} \tag{15.43}$$

После того как вычисления по методу обратного распространения выполнены до момента времени $n - h+1$, происходит следующая коррекция синаптических весов w_{ji} нейрона j:

$$\Delta w_{ji}(n) = \eta \sum_{l=n-h+1}^{n} \delta_j(l)x_i(l-1), \tag{15.44}$$

где η и $x_i(l-1)$ уже были определены ранее. Обратите внимание, что использование $w_{kj}(l)$ в выражении (15.43) требует, чтобы поддерживалась некоторая история значений весов. Использование величин w_{kj} в этом выражении обосновано только в том случае, когда параметр скорости обучения η достаточно мал для того, чтобы значения весов кардинально не изменялись при переходе от одного шага времени к следующему.

При сравнении выражений (15.43) и (15.40) видно, что, в отличие от алгоритма BPTT, в вычислении присутствует только сигнал ошибки, вычисленный в текущий момент времени n. Это и объясняет причину, по которой не сохраняются прошлые значения желаемых откликов. В результате усеченный алгоритм обратного распространения во времени учитывает вычисления всех предыдущих шагов, аналогично тому, как стохастический алгоритм обратного распространения (см. главу 4) учитывает вычисления, производимые скрытыми нейронами в многослойном персептроне.

Некоторые практические соглашения

На практике в приложениях алгоритма BPTT использование усечения оказывается не таким уж искусственным, как это казалось на первый взгляд. Если рекуррентная есть не является неустойчивой, должна обеспечиваться сходимость производных $\partial \mathbf{E}(l)/\partial v_j(l)$, так как обратный отсчет по времени соответствует увеличению мощности сигнала обратной связи (приблизительно равному наклону сигмоидной функции,

умноженному на веса). В любом случае глубина усечения h должна быть достаточно большой, чтобы вычисляемые производные были близки к реальным значениям. Например, в приложении динамически управляемых рекуррентных сетей к управлению холостым ходом двигателя рассматривается значение h =30, которое является довольно консервативным для выполнения задачи обучения [863].

Следует обсудить еще один практический вопрос. Процедура развертывания для обратного распространения во времени, описанная в этом разделе, дает полезный инструмент ее описания в терминах каскада идентичных слоев, ниспадающих в направлении течения времени, что обеспечивает ключ к пониманию ее работы. Однако эта сильная точка является одновременно и слабой. Эта процедура прекрасно работает в относительно простых рекуррентных сетях, состоящих из малого числа нейронов. Однако рассматриваемые формулы (особенно (15.43)) становятся огромными, когда процедура развертывания применяется к более общим архитектурам, которые, как правило, и встречаются на практике. В таких ситуациях предпочтительнее использовать более общий подход, описанный в [1126]. В нем каждое выражение прямого распространения слоя приводит к увеличению соответствующего множества выражений обратного распространения. Достоинством этого подхода является гомогенное толкование прямых и рекуррентных (обратных) связей.

Для того чтобы описать принцип работы этой частной формы алгоритма BPTT(h), обозначим символом F^l_{-x} *упорядоченную производную* (ordered derivative) выхода сети в узле l по x. Для вывода уравнений обратного распространения уравнения прямого распространения рассматриваются в обратном порядке. Из каждого уравнения выводится одно или несколько выражений обратного распространения в соответствии со следующим принципом:

$$\text{Если } a = \varphi(b, c), \text{ то } F^l_{-b} = \frac{\partial \varphi}{\partial b} F^l_{-a} \text{ и } F^l_c = \frac{\partial \varphi}{\partial c} F^l_{-a}. \tag{15.45}$$

Пример 15.5

Для того чтобы прояснить значение понятия упорядоченных производных, рассмотрим нелинейную систему, описываемую следующей системой уравнений:

$$x_1 = \log u + x_2^3,$$
$$y = x_1^2 + 3x_2.$$

Переменная x_2 влияет на выход y двояким образом: непосредственно из второго уравнения и неявно через первое уравнение. Упорядоченная производная y по x_2 определяется общим влиянием, которое учитывает прямой и неявный эффекты:

$$F_{-x_2} = \frac{\partial y}{\partial x_2} + \frac{\partial y}{\partial x_1}\frac{\partial x_1}{\partial x_2} = 3 + (2x_1)(3x_2^2) = 3 + 6x_1x_2^2.$$

■

При программировании упорядоченных производных для BPTT(h) величины в правой части каждой из упорядоченных производных (15.45) складываются с предыдущим значением левой части. Таким образом, соответствующие производные распределяются от данного узла ко всем узлам и синаптическим весам, которые передают информацию данному узлу (т.е. в обратном направлении), с учетом единичных задержек, которые могут иметь место в таких связях. Простота описанной здесь формулировки избавляет от необходимости визуализации, такой как развертывание во времени или граф передачи сигнала. В [291], [863] эта процедура использовалась для получения псевдокода реализации алгоритма PBTT(h).

15.8. Рекуррентное обучение в реальном времени

В этом разделе описывается еще один алгоритм обучения, называемый *рекуррентным обучением в реальном времени* (real-time recurrent learning — RTRL(h))[9]. Этот алгоритм получил свое название из-за того, что коррекция синаптических весов полносвязной рекуррентной сети выполняется в реальном времени, т.е. непосредственно в ходе выполнения сетью своих функций обработки сигнала [1157]. На рис. 15.11 показана структура такой рекуррентной сети. Она состоит из q нейронов с m внешними входами. Эта сеть имеет два различных слоя: *конкатенированного слоя "вход–обратная связь"* (concatenated input-feedback layer) и *слоя вычислительных узлов* (processing layer of computational nodes). Соответственно синаптические связи сети делятся на прямые и обратные.

Описание этой сети в пространстве состояний дается уравнениями (15.10) и (15.11). Уравнение процесса (15.10) воспроизведем в следующей расширенной форме:

$$\mathbf{x}(n+1) = \begin{bmatrix} \varphi(\mathbf{w}_1^T \boldsymbol{\xi}(n)) \\ \cdots \\ \varphi(\mathbf{w}_j^T \boldsymbol{\xi}(n)) \\ \cdots \\ \varphi(\mathbf{w}_q^T \boldsymbol{\xi}(n)) \end{bmatrix}, \tag{15.46}$$

где предполагается, что все нейроны имеют одну и ту же функцию активации $\varphi(\cdot)$. Вектор $\mathbf{w}_j$ размерности $(q+m+1)\times 1$ является вектором синаптических весов отдель-

[9] Алгоритм рекуррентного обучения в реальном времени впервые был описан в литературе по нейронным сетям в [1157]. Его происхождение можно отследить до ранней работы, написанной в 1965 году и посвященной системной идентификации для настройки параметров произвольных динамических систем [711].

Вывод для одного слоя вполне рекуррентных нейронов был представлен в [1157], впоследствии он был расширен на более общие архитектуры [552], [861].

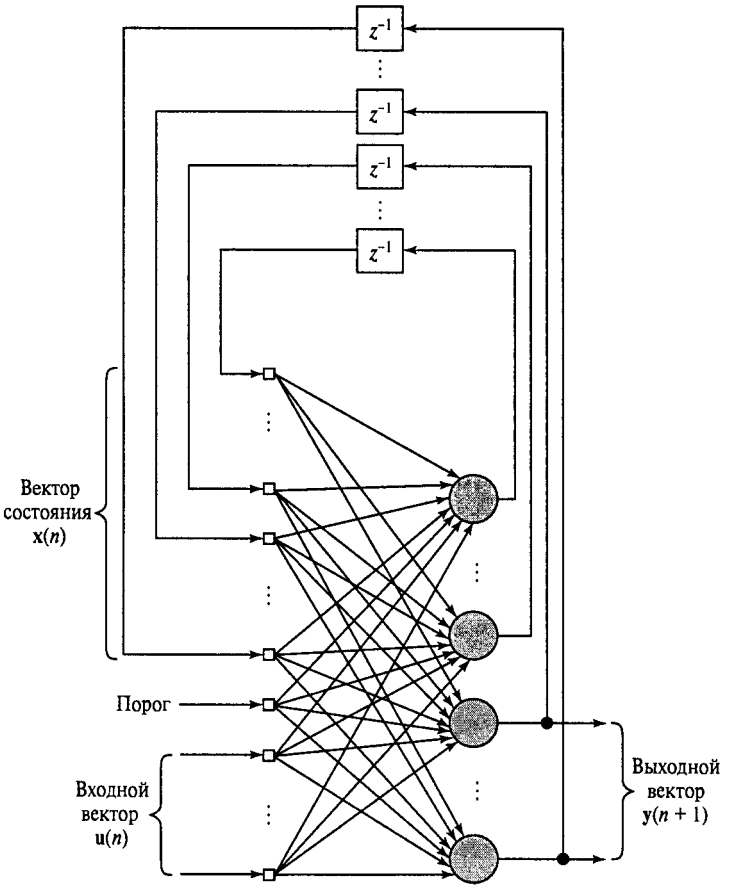


Рис. 15.11. Полносвязная рекуррентная сеть для алгоритма RTRL

ного нейрона j рекуррентной сети, т.е.

$$\mathbf{w}_j = \begin{bmatrix} \mathbf{w}_{a,j} \\ \mathbf{w}_{b,j} \end{bmatrix}, \quad j = 1, 2, \ldots, q, \tag{15.47}$$

где $\mathbf{w}_{a,j}$ и $\mathbf{w}_{b,j}$ — j-е столбцы транспонированных матриц весов $\mathbf{W}_a^T$ и $\mathbf{W}_b^T$ соответственно. Вектор $\boldsymbol{\xi}(n)$ размерности $(q + m{+}1){\times}1$ определяется следующим образом:

$$\boldsymbol{\xi}(n) = \begin{bmatrix} \mathbf{x}(n) \\ \mathbf{u}(n) \end{bmatrix}, \tag{15.48}$$

где $\mathbf{x}(n)$ — вектор состояния размерности $q{\times}1$; $\mathbf{u}(n)$ — входной вектор размерности $(m{+}1){\times}1$. Первым элементом вектора $\mathbf{u}(n)$ является число +1, и соответственно первый элемент вектора $\mathbf{w}_{b,j}$ равен внешнему смещению b_j, применяемому к нейрону j.

Для упрощения выкладок мы введем новые матрицы $\mathbf{\Lambda}_j(n)$, $\mathbf{U}_j(n)$ и $\mathbf{\Phi}(n)$ следующего вида.

1. $\mathbf{\Lambda}_j(n)$ — матрица размерности $q \times (q + m + 1)$, определяемая как частная производная вектора состояния $\mathbf{x}(n)$ по вектору весов $\mathbf{w}_j$:

$$\mathbf{\Lambda}_j(n) = \frac{\partial \mathbf{x}(n)}{\partial \mathbf{w}_j}, \; j = 1, 2, \ldots, q. \tag{15.49}$$

2. $\mathbf{U}_j(n)$ — матрица размерности $q \times (q+m+1)$, все строки которой, за исключением строки j, равны нулю. Строка j равна транспонированному вектору $\boldsymbol{\xi}(n)$:

$$\mathbf{U}_j(n) = \begin{bmatrix} 0 \\ \boldsymbol{\xi}^T(n) \\ 0 \end{bmatrix} \leftarrow j\text{-я строка}, \quad j = 1, 2, \ldots, q. \tag{15.50}$$

3. $\mathbf{\Phi}(n)$ — диагональная матрица размерности $q \times q$, j-й диагональный элемент которой является частной производной функции активации по своему аргументу, вычисленному в точке $\mathbf{w}_j^T \boldsymbol{\xi}(n)$:

$$\mathbf{\Phi}(n) = \operatorname{diag}\left(\varphi'(\mathbf{w}_1^T \boldsymbol{\xi}(n)), \ldots, \varphi'(\mathbf{w}_j^T \boldsymbol{\xi}(n)), \ldots, \varphi'(\mathbf{w}_q^T \boldsymbol{\xi}(n))\right). \tag{15.51}$$

Учитывая эти определения, мы можем продифференцировать уравнение (15.46) по $\mathbf{w}_j$. Тогда, используя *цепное правило* (chain rule of calculus), получим следующее рекурсивное уравнение:

$$\mathbf{\Lambda}_j(n+1) = \mathbf{\Phi}(n) \left[\mathbf{W}_a(n) \mathbf{\Lambda}_j(n) + \mathbf{U}_j(n)\right], \quad j = 1, 2, \ldots, q. \tag{15.52}$$

Это рекурсивное уравнение описывает *нелинейную динамику состояний* (nonlinear state dynamics) (т.е. эволюцию состояний) процесса рекуррентного обучения в реальном времени.

Для того чтобы завершить описание этого процесса обучения, нужно связать матрицу $\mathbf{\Lambda}_j(n)$ с градиентом поверхности ошибок по $\mathbf{w}_j$. Для этого сначала используем уравнение получения измерения (15.11) и определим вектор ошибки размерности $p \times 1$:

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{y}(n) = \mathbf{d}(n) - \mathbf{C}\mathbf{x}(n). \tag{15.53}$$

Мгновенная сумма квадратичной ошибки в момент времени n определяется в терминах $\mathbf{e}(n)$ следующим образом:

$$\mathbf{E}(n) = \frac{1}{2}\mathbf{e}^T(n)\mathbf{e}(n). \tag{15.54}$$

Цель процесса обучения — минимизация функции стоимости, полученной суммированием величин $\mathbf{E}(n)$ по всем моментам времени n, т.е.

$$\mathbf{E}_{\text{общ.}} = \sum_n \mathbf{E}(n).$$

Для достижения этой цели можно использовать метод наискорейшего спуска, который требует знания *матрицы градиентов* (gradient matrix), которая определяется следующим образом:

$$\nabla_{\mathbf{w}}\mathbf{E}_{\text{общ.}} = \frac{\partial \mathbf{E}_{\text{общ.}}}{\partial \mathbf{W}} = \sum_n \frac{\partial \mathbf{E}(n)}{\partial \mathbf{W}} = \sum_n \nabla_{\mathbf{w}}\mathbf{E}(n),$$

где $\nabla_{\mathbf{w}}\mathbf{E}(\mathbf{n})$ — градиент $\mathbf{E}(n)$ по отношению к матрице весов $\mathbf{W}=\{\mathbf{w}_k\}$. При желании можно продолжить работу с этим уравнением и вывести уравнения коррекции для синаптических весов рекуррентной сети, не прибегая к аппроксимациям. Однако, для того чтобы получить алгоритм обучения, который можно использовать для обучения рекуррентной сети *в реальном времени*, необходимо использовать мгновенную *оценку* градиента, а именно $\nabla_{\mathbf{w}}\mathbf{E}(n)$, что приведет к приближению метода наискорейшего спуска.

Возвращаясь к формуле (15.54), описывающей минимизируемую функцию стоимости, продифференцируем ее по вектору весов $\mathbf{w}_j$ и получим следующее:

$$\frac{\partial \mathbf{E}(n)}{\partial \mathbf{w}_j} = \left(\frac{\partial \mathbf{e}(n)}{\partial \mathbf{w}_j}\right)\mathbf{e}(n) = -\left(\frac{\partial \mathbf{x}(n)}{\partial \mathbf{w}_j}\right)\mathbf{e}(n) = -\mathbf{C}\boldsymbol{\Lambda}_j(n)\mathbf{e}(n), j = 1, 2, \ldots, q. \tag{15.55}$$

Исходя из этого, коррекция, применяемая к вектору синаптических весов $\mathbf{w}_j(n)$ нейрона j, может быть определена как

$$\Delta\mathbf{w}_j(n) = -\eta\frac{\partial \mathbf{E}(n)}{\partial \mathbf{w}_j} = \eta\mathbf{C}\boldsymbol{\Lambda}_j(n)\mathbf{e}(n), \quad j = 1, 2, \ldots, q, \tag{15.56}$$

где η — параметр скорости обучения, а $\lambda_j(n)$ определяется формулой (15.52).

ТАБЛИЦА 15.1. Алгоритм рекуррентного обучения в реальном времени

Параметры

m — размерность входного пространства

q— размерность пространства состояний

p— размерность выходного пространства

$\mathbf{w}_j$ — вектор синаптических весов нейрона $j, j = 1, 2, \ldots, q$

Инициализация

1. Устанавливаем синаптические веса сети в малые значения, выбираемые из равномерного распределения.
2. Устанавливаем начальное значение вектора состояний $\mathbf{x}(0)$ в $\mathbf{0}$.
3. Присваиваем $\mathbf{\Lambda}_j(0) = \mathbf{0}$ для $j = 1, 2, \ldots, q$

Вычисления

Для n =1, 2, ..., q вычисляем

$$\mathbf{\Lambda}_j(n+1) = \mathbf{\Phi}(n)\left[\mathbf{W}_a(n)\mathbf{\Lambda}_j(n) + \mathbf{U}_j(n)\right],$$
$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{C}\mathbf{x}(n),$$
$$\Delta\mathbf{w}_j(n) = \eta\mathbf{C}\mathbf{\Lambda}_j(n)\mathbf{e}(n).$$

Определения величин $\mathbf{x}(n)$, $\mathbf{\Lambda}_j(n)$, $\mathbf{U}_j(n)$ и $\mathbf{\Phi}(n)$ заданы формулами (15.46), (15.49)–(15.51) соответственно.

Единственным вопросом остается определение *начальных условий* (initial conditions) для запуска процесса обучения. С этой целью положим

$$\mathbf{\Lambda}_j(0) = \mathbf{0} \text{ для всех } j. \tag{15.57}$$

Это значит, что изначально рекуррентная сеть находится в постоянном состоянии.

В табл. 15.1 в сжатом виде приведен алгоритм рекуррентного обучения в реальном времени. Описанная здесь формулировка этого алгоритма применима к любой функции активации $\varphi(\cdot)$, которая дифференцируема по своему аргументу. Для частного случая сигмоидной нелинейности в форме гиперболического тангенса имеем:

$$x_j(n+1) = \varphi(v_j(n)) = \tanh(v_j(n))$$

и

$$\varphi'(v_j(n)) = \frac{\partial\varphi(v_j(n))}{\partial v_j(n)} = \operatorname{sech}^2(v_j(n)) = 1 - x_j^2(n+1), \tag{15.58}$$

где $v_j(n)$ — индуцированное локальное поле нейрона j; $x_j(n+1)$ — его состояние в момент времени $n+1$.

Использование мгновенного градиента $\nabla_{\mathbf{w}}\mathbf{E}(n)$ означает, что описанный здесь алгоритм рекуррентного обучения в реальном времени отклоняется от алгоритма не в реальном времени, основанного на истинном градиенте $\nabla_{W}\mathbf{E}_{\text{общ.}}$. Тем не менее это отклонение в точности аналогично отклонению, обнаруженному в стандартном алгоритме обратного распространения, использовавшемся в главе 4 для обучения многослойного персептрона, в котором коррекция весов проводилась после подачи в сеть каждого примера. В то время как алгоритм рекуррентного обучения в реальном времени не гарантирует точного следования направлению, противоположному направлению градиента общей функции ошибок $\mathbf{E}_{\text{общ.}}(\mathbf{W})$ по отношению к матрице весов $\mathbf{W}$, практическое отличие между версиями реального и не реального времени зачастую незначительны; эти две версии практически идентичны при достаточно малых значениях параметра скорости обучения η. Более серьезное последствие этого отклонения от истинно градиентной динамики состоит в том, что наблюдаемая траектория (получаемая как график $\mathbf{E}(n)$ относительно элементов матрицы весов $\mathbf{W}(n)$) может сама зависеть от коррекции весов, производимой алгоритмом, что можно рассматривать как еще один источник обратной связи, который может привести к неустойчивости системы. Этот эффект можно свести на нет, приняв параметр скорости обучения настолько малым, чтобы шкала времени коррекции весов была значительно меньшей, чем шкала времени реальной работы сети [1157].

Пример 15.6

В этом примере сформулируем алгоритм RTRL для *полностью рекуррентной сети* (fully recurrent network), показанной на рис. 15.6 и имеющей два входа и один выход. Эта сеть имеет три нейрона и композицию матриц $\mathbf{W}_a$, $\mathbf{W}_b$ и $\mathbf{C}$, которые были описаны в примере 15.1.

При $m = 2$ и $q = 3$ в (15.48) обнаруживаем, что

$$\boldsymbol{\xi}(n) = \begin{bmatrix} x_1(n) \\ x_2(n) \\ x_3(n) \\ 1 \\ u_1(n) \\ u_2(n) \end{bmatrix}.$$

Пусть kl-й элемент матрицы $\boldsymbol{\Lambda}_j(n)$ имеет обозначение $\lambda_{j,kl}(n)$. Тогда, подставляя выражения (15.52) и (15.56), получим:

$$\lambda_{j,kl}(n+1) = \varphi'(v_j(n)) \left[\sum_{i=1}^{3} w_{ji}(n)\lambda_{j,kl}(n) + \delta_{kj}\xi_l(n) \right],$$

$$\Delta w_{kl}(n) = \eta(d_1(n)) - x_1(n)\lambda_{1,kl}(n),$$

где δ_{kj} — дельта Кронекера, которая равна единице при $k = j$ и нулю в противном случае, и $(j,k) = 1, 2, 3$ и $l = 1, 2, \ldots, 6$. На рис. 15.12 показан *граф чувствительности*, показывающий эволюцию коррекции весов $\Delta w_{kl}(n)$. Обратите внимание, что $\mathbf{W}_a = \{w_{ji}\}$ для $(j,i) = 1,2,3$ и $\mathbf{W}_b = \{w_{jl}\}$ для $j = 1,2,3$ и $l = 4,5,6$. ■

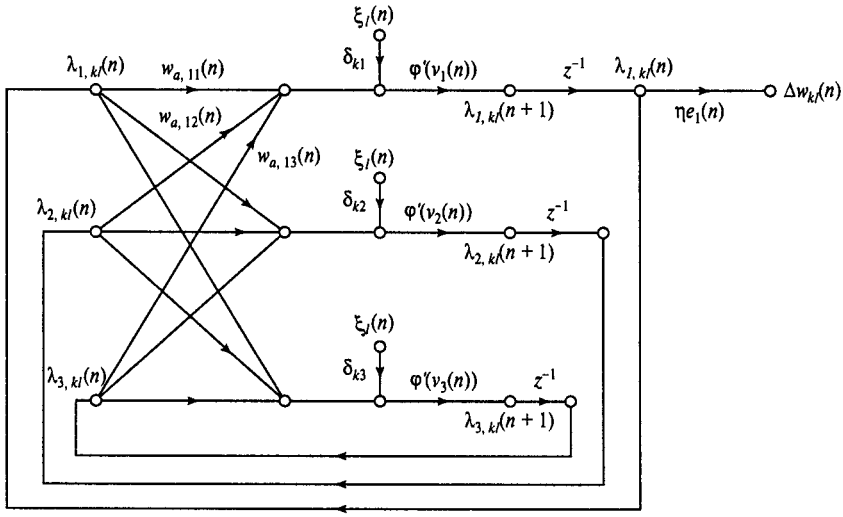


Рис. 15.12. Граф чувствительности полностью рекуррентной сети, показанной на рис. 15.6. Примечание: три узла с метками $\xi_l(n)$ следует рассматривать как единый вход

Усиление учителем

Одна из стратегий, которые используются при обучении рекуррентных сетей, называется *усиление учителем* (teacher forcing) [1156], [1157]. В адаптивной фильтрации усиление учителем известно как *метод уравнения ошибок* (equation-error method) [725]. В своей основе усиление учителем использует замещение во время обучения реального выходного сигнала сети соответствующим желаемым откликом (т.е. целевым сигналом) при последующем вычислении динамики сети (когда этот желаемый отклик доступен). Несмотря на то что метод усиления учителем описывается при рассмотрении алгоритма RTRL, он применим и к любому другому алгоритму обучения. Однако для его применяемости рассматриваемый нейрон должен замыкать свой выход в обратную связь.

Среди преимуществ метода усиления учителем можно выделить следующие [1156].

- *Усиление учителем может привести к ускорению обучения.* Причина этого улучшения — в использовании величин усиления учителем в качестве предположения, что сеть корректно обучена всем более ранним частям задачи, относящимся к нейронам, к которым применяется усиление учителем.
- *Усиление учителем может служить механизмом коррекции при обучении.* Например, синаптические веса сети могут иметь корректные значения, но сеть временно может работать в несвойственных ей областях пространства состояний. Естественно, в такой ситуации коррекция синаптических весов была бы неправильной стратегией.

Градиентные алгоритмы обучения, которые используют усиление учителем, на самом деле оптимизируют функцию стоимости по-другому, нежели их неусиленные аналоги. Таким образом, усиленные и неусиленные учителем версии алгоритма могут давать различные решения (если соответствующие сигналы ошибки не равны нулю, то обучение вообще не требуется).

15.9. Фильтр Калмана

Как уже говорилось, непрерывное обучение, основанное на градиентном спуске (показанное на примере алгоритма рекуррентного обучения в реальном времени), осуществляется довольно медленно из-за доверия к мгновенным оценкам градиентов. Это серьезное ограничение можно обойти, рассмотрев обучение рекуррентной сети с учителем как *задачу оптимальной фильтрации* (optimum filtering problem), решение которой *рекурсивно* использует информацию, содержащуюся в данных обучения, неявно возвращаясь к первой итерации процесса обучения. Описанная здесь идея лежит в основе *фильтрации Калмана* (Kalman filtering) [535]. Среди новаторских признаков фильтра Калмана следует выделить следующие.

- Эта теория сформулирована в терминах концепций пространства состояний и предлагает эффективное использование информации, содержащейся во входных данных.
- Оценка состояния выполняется рекурсивно. Это значит, что каждая скорректированная оценка состояния вычисляется на основе предыдущей оценки и доступных в текущий момент данных. Следовательно, сохранение требуется только для предыдущей оценки.

В этом разделе представлен краткий обзор теории фильтров Калмана[10]. Этим мы проложим путь для вывода в следующем разделе несвязного расширенного фильтра Калмана. Развитие этой теории, как обычно, началось с линейных динамических систем. Для того чтобы расширить ее использование на нелинейные динамические системы, к системе применяется одна из форм *линеаризации*. Последний вопрос мы отложим до следующего раздела.

Итак, рассмотрим *линейную, дискретную динамическую систему*, показанную на рис. 15.13. Временное описание представленной здесь системы сохраняет сходство с формализмом пространства состояний, представленным в разделе 15.3. В математических терминах на рис. 15.13 реализована следующая система уравнений:

---

[10] Теория фильтров Калмана берет свое начало в классической работе [535]. Эта теория заняла свое место как существенная часть управления и обработки сигнала и нашла применение в разнообразных областях деятельности человека. Подробное описание стандартного фильтра Калмана, его вариантов и расширенных форм, работающих с нелинейными динамическими системами, содержится в [385], [434]. Первая работа целиком посвящена теории и практике фильтрации Калмана, а во второй рассматривается теория фильтров Калмана с позиций адаптивной фильтрации. Среди других книг, посвященных этому вопросу, можно выделить [513], [708], [709].

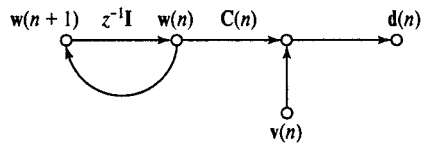


Рис. 15.13. Граф передачи сигнала линейной дискретной динамической системы, используемой для описания фильтра Калмана

$$\mathbf{w}(n+1) = \mathbf{w}(n), \tag{15.59}$$

$$\mathbf{d}(n) = \mathbf{C}(n)\mathbf{w}(n) + \mathbf{v}(n). \tag{15.60}$$

Величины, использованные в *уравнении процесса* (15.59) и *уравнении измерения* (15.60), можно описать следующим образом.

- $\mathbf{w}(n)$ — *вектор состояния* (state vector) системы.
- $\mathbf{d}(n)$ — *вектор наблюдения* (observation vector).
- $\mathbf{C}(n)$ — *матрица измерений* (measurement matrix).
- $\mathbf{v}(n)$ — *шум или погрешность измерений* (measurement noise).

В уравнении процесса (15.59) сделаны два упрощающих допущения. Во-первых, уравнение процесса *не учитывает шума*. Во-вторых, передаточная матрица, связывающая состояния системы в моменты времени $n+1$ и n, равна единичной. На рис. 15.13 также использовалось новое понятие состояния (по причинам, которые станут понятны в следующем разделе).

Задача фильтрации Калмана может быть сформулирована следующим образом.

Используя все множество наблюдаемых данных, состоящее из набора векторов $\{\mathbf{d}(i)\}_{i=1}^{n}$, *нужно найти для каждого* $n \geq 1$ *оценку с минимальной среднеквадратической ошибкой состояния* $\mathbf{w}(i)$.

Обратите внимание, что информация о векторе состояний недоступна. Эта задача называется *задачей фильтрации* (при $i = n$), *прогнозирования* (при $i > n$) или *сглаживания* (при $1 \leq i \leq n$). Решение этой задачи получается на основе следующих допущений.

1. Погрешность измерений $\mathbf{v}(n)$ имеет нулевое среднее значение и является процессом белого шума с матрицей ковариации:

$$E[\mathbf{v}(n)\mathbf{v}^T(k)] = \begin{cases} \mathbf{R}(n), & n = k, \\ \mathbf{0}, & n \neq k. \end{cases} \tag{15.61}$$

2. Начальное состояние $\mathbf{w}(0)$ не коррелировано с $\mathbf{v}(n)$ для всех $n \geq 0$.

Для изящного вывода фильтра Калмана можно использовать понятие инновации [534]. Более точно, *процесс инновации* (innovation process), связанный с вектором наблюдений $\mathbf{d}(n)$, определяется следующим образом:

$$\boldsymbol{\alpha}(n) = \mathbf{d}(n) - \hat{\mathbf{d}}(n|n-1), \tag{15.62}$$

где $\hat{\mathbf{d}}(n|n-1)$ — *оценка* $\mathbf{d}(n)$ *с минимальной среднеквадратической ошибкой* на основе *всех прошлых* значений вектора наблюдений, начиная с момента времени $n = 1$ и заканчивая моментом $n-1$. Под "оценкой с минимальной среднеквадратической ошибкой" понимается та оценка, которая минимизирует среднеквадратическую ошибку $\mathbf{d}(n)$. Инновационный процесс $\boldsymbol{\alpha}(n)$ можно рассматривать как *меру новой информации*, содержащейся в $\mathbf{d}(n)$, которая была недоступна в прогнозируемой части $\hat{\mathbf{d}}(n|n-1)$. Инновационный процесс имеет ряд следующих полезных свойств [534].

1. Инновационный процесс $\boldsymbol{\alpha}(n)$, ассоциированный с $\mathbf{d}(n)$, не коррелирован со всеми предыдущими наблюдениями $\mathbf{d}(1)$, $\mathbf{d}(2)$,..., $\mathbf{d}(n-1)$, т.е.

$$E[\boldsymbol{\alpha}(n)\mathbf{d}^T(k)] = \mathbf{0} \text{ для } 1 \leq k \leq n-1.$$

2. Инновационный процесс состоит из последовательности случайных векторов, которые не коррелированы друг с другом:

$$E[\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^T(k)] = \mathbf{0} \text{ для } 1 \leq k \leq n-1.$$

3. Существует однозначное соответствие между последовательностью случайных векторов, представляющих наблюдаемые данные, и последовательностью случайных векторов, представляющих инновационный процесс:

$$\{\mathbf{d}(1), \mathbf{d}(2), \ldots, \mathbf{d}(n)\} \rightleftharpoons \{\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \ldots, \boldsymbol{\alpha}(n)\}. \tag{15.63}$$

Теперь можно заменить коррелированную последовательность наблюдаемых данных некоррелированной (и, следовательно, упрощенной) последовательностью инноваций, при этом *не теряя информацию*. Это упрощает вывод фильтра Калмана, так как выражает оценку состояния в момент времени i через заданный набор инноваций $\{\boldsymbol{\alpha}(k)\}_{k=1}^{n}$. При проведении этого анализа можно вывести стандартный фильтр Калмана (табл. 15.2).

В этом алгоритме существуют три новые величины, которые требуют определения.

ТАБЛИЦА 15.2. Фильтр Калмана

Для $n = 1, 2, \ldots$ вычисляем:

$$\begin{aligned}
\boldsymbol{\Gamma}(n) &= [\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^T(n) + \mathbf{R}(n)]^{-1}, \\
\mathbf{G}(n) &= \mathbf{K}(n, n-1)\mathbf{C}^T(n)\boldsymbol{\Gamma}(n), \\
\boldsymbol{\alpha}(n) &= \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{w}}(n|n-1), \\
\hat{\mathbf{w}}(n+1|n) &= \hat{\mathbf{w}}(n|n-1) + \mathbf{G}(n)\boldsymbol{\alpha}(n), \\
\mathbf{K}(n+1, n) &= \mathbf{K}(n, n-1) - \mathbf{G}(n)\mathbf{C}(n)\mathbf{K}(n, n-1)
\end{aligned}$$

- $\mathbf{K}(n, n-1)$ — *матрица ковариации ошибки* (error covariance matrix), определяемая следующим образом:

$$\mathbf{K}(n, n-1) = E[\boldsymbol{\varepsilon}(n, n-1)\boldsymbol{\varepsilon}^T(n, n-1)], \tag{15.64}$$

  где *вектор состояния* $\varepsilon(n, n-1)$, в свою очередь, определяется как

$$\boldsymbol{\varepsilon}(n, n-1) = \mathbf{w}(n) - \hat{\mathbf{w}}(n|n-1), \tag{15.65}$$

  где $\mathbf{w}(n)$ — фактическое состояние; $\hat{\mathbf{w}}(n|n-1)$ — его прогнозируемое значение на один шаг вперед, основанное на предыдущих значениях данных наблюдения вплоть до момента времени $n-1$.
- $\boldsymbol{\Gamma}(n)$ — *переводной коэффициент* (conversion factor), который связывает *ошибку фильтрованной оценки* (filtered estimation error) $\mathbf{e}(n)$ с инновацией $\boldsymbol{\alpha}(n)$:

$$\mathbf{e}(n) = \mathbf{R}(n)\boldsymbol{\Gamma}(n)\boldsymbol{\alpha}(n), \tag{15.66}$$

  где

$$\mathbf{e}(n) = \mathbf{d}(n) - \hat{\mathbf{d}}(n|n) \tag{15.67}$$

  и $\hat{\mathbf{d}}(n|n)$ — оценка вектора наблюдений $\mathbf{d}(n)$, полученная на основе всех наблюдений вплоть до момента времени n.
- $\mathbf{G}(n)$ — *коэффициент усиления Калмана* (Kalman gain), который определяет коррекцию, применяемую к оценке состояния.

Тип фильтра Калмана, приведенного в табл. 15.2, предназначен для распространения матрицы ковариации ошибки $\mathbf{K}(n, n-1)$ и поэтому называется *ковариационным алгоритмом фильтрации Калмана* (covariance Kalman filtering algorithm).

Фильтр Калмана на основе квадратного корня

Ковариационный фильтр Калмана связан с серьезными вычислительными проблемами. В частности, обновленная матрица $\mathbf{K}(n+1, n)$ определяется *уравнением Рикатти* (Ricatti equation) (см. последнюю строку в табл. 15.2). В правой части уравнения Рикатти содержится разность между двумя матричными величинами. Если числовая точность на каждом шаге итерации алгоритма не будет достаточно велика, то скорректированная матрица $\mathbf{K}(n+1, n)$ может *не* оказаться неотрицательно определенной. Понятно, что такое решение неприемлемо, так как $\mathbf{K}(n+1, n)$ представляет собой матрицу ковариации, которая неотрицательно определена *по определению*. Такая неустойчивая динамика фильтра Калмана, ставшая результатом неточности вычислений, связана с использованием арифметики с конечной длиной представления и называется *феноменом дивергенции* (divergence phenomenon).

Этой проблемы можно избежать, распространяя квадратный корень матрицы ковариации ошибки $\mathbf{K}^{1/2}(n, n-1)$, а не саму матрицу $\mathbf{K}(n, n-1)$. В частности, используя *разложение Холецкого* (Cholesky factorization), $\mathbf{K}(n, n-1)$ можно выразить следующим образом [368]:

$$\mathbf{K}(n, n-1) = \mathbf{K}^{1/2}(n, n-1)\mathbf{K}^{T/2}(n, n-1), \tag{15.68}$$

где $\mathbf{K}^{1/2}(n, n-1)$ — нижняя треугольная матрица; $\mathbf{K}^{T/2}(n, n-1)$ — транспонированная к ней. В линейной алгебре множитель Холецкого $\mathbf{K}^{1/2}(n, n-1)$ называют квадратным корнем матрицы $\mathbf{K}(n, n-1)$. Поэтому фильтр Калмана, основанный на разложении Холецкого, принято называть *фильтром Калмана на основе квадратного корня*[11]. Здесь важно отметить, что произведение матриц $\mathbf{K}^{1/2}(n, n-1)\mathbf{K}^{T/2}(n, n-1)$ менее вероятно может стать неопределенным, так как произведение любой квадратной матрицы на себя же транспонированную всегда положительно определено.

15.10. Несвязный расширенный фильтр Калмана

В фильтре Калмана нас прежде всего интересуют те его уникальные свойства, которые можно использовать для обучения рекуррентной сети[12]. Для рекуррентной

[11] Детальное описание фильтра Калмана на основе квадратного корня и эффективные методы его реализации содержатся в [434].

[12] Первой работой, в которой продемонстрирована повышенная производительность нейронных сетей, обучаемых с учителем и использующих расширенный фильтр Калмана, была, пожалуй, [997]. К сожалению, описанный в ней алгоритм обучения имеет свои ограничения, так как чрезвычайно сложен с вычислительной точки зрения. Чтобы обойти это ограничение, в [588], [967] была предпринята попытка упростить применение расширенной фильтрации Калмана за счет деления глобальной задачи на ряд подзадач, каждая из которых разрешалась одним нейроном. Однако толкование каждого нейрона как задачи идентификации не было строго связано с теорией фильтров Калмана. Более того, такой подход мог привести к неустойчивости в процессе обучения, что, в свою очередь, приводило к получению худшего решения, нежели получаемое другими методами [864].

сети заданной архитектурной сложности (например, рекуррентного многослойного персептрона) важным вопросом является облегчение вычислений без ущерба для применения теории фильтров Калмана. Ответ на этот вопрос лежит в использовании *несвязной* формы расширенного фильтра Калмана, в которой вычислительная сложность приводится в соответствие с требованиями конкретного приложения и доступными вычислительными ресурсами [864].

Рассмотрим рекуррентную сеть, построенную на основе статического многослойного персептрона с W синаптическими весами и p выходными узлами. Пусть в векторе $\mathbf{w}(n)$ содержатся все синаптические веса сети в момент времени n. Подразумевая адаптивные фильтры, уравнение этой сети в пространстве состояний можно промоделировать следующим образом [434], [997]:

$$\mathbf{w}(n+1) = \mathbf{w}(n), \tag{15.69}$$

$$\mathbf{d}_o(n) = \mathbf{c}(\mathbf{w}(n), \mathbf{u}(n), \mathbf{v}(n)) + \mathbf{v}(n), \tag{15.70}$$

где вектор весов $\mathbf{w}(n)$ играет роль состояния. Второй аргумент, $\mathbf{u}(n)$, и третий аргумент, $\mathbf{v}(n)$, вектор-функции $\mathbf{c}(\cdot,\cdot,\cdot)$ обозначают соответственно входной вектор и вектор рекуррентной активности узла. В результате уравнение (15.69) утверждает, что система находится в "оптимальном" состоянии, в котором переходная матрица, преобразовывающая вектор весов $\mathbf{w}(n)$ из момента времени n в вектор весов $\mathbf{w}(n+1)$ момента времени $n+1$, равна единичной матрице. Описанное здесь оптимальное состояние соответствует локальному или глобальному минимуму на поверхности ошибок рекуррентной сети. Единственный источник нелинейности в этой системе находится в уравнении измерения (15.70). Вектор $\mathbf{d}_o$ обозначает желаемый отклик модели. Так как уравнение (15.70) представляет собой отношение выходного сигнала к входному в модели, то отсюда следует, что $\mathbf{c}(\cdot,\cdot,\cdot)$ — общая нелинейность, связывающая входной и выходной слой многослойного персептрона. Предполагается, что вектор погрешности измерений $\mathbf{v}(n)$ в (15.70) является процессом белого шума, зависящего от многих переменных, с нулевым средним и матрицей ковариации $\mathbf{R}(n)$.

Следует заметить, что при применении расширенного фильтра Калмана в рекуррентной сети существуют два контекста использования понятия "состояние".

- Эволюция системы при адаптивной фильтрации, которая проявляется в изменении весов рекуррентной сети в процессе обучения. Это первое значение состояния берет на себя вектор $\mathbf{w}(n)$.
- Работа самой рекуррентной сети, показанная на примере рекуррентного узла, от которого зависит функция $\mathbf{c}$. Это значение состояния берет на себя вектор $\mathbf{v}(n)$.

Сравнивая модель, описываемую системой (15.69) и (15.70), с линейной динамической моделью (15.59) и (15.60), видим, что единственным отличием между ними является нелинейность формы уравнения измерения. Чтобы подготовить почву для применения теории фильтров Калмана к описанной модели в пространстве состояний, необходимо вначале линеаризовать уравнение (15.70) и представить его в виде

$$\mathbf{d}(n) = \mathbf{C}(n)\mathbf{w}(n) + \mathbf{v}(n), \tag{15.71}$$

где $\mathbf{C}(n)$ — матрица измерения линеаризованной модели размерности $p \times W$; для отличия от уравнения (15.70) вместо $\mathbf{d}_o(n)$ использовалось обозначение $\mathbf{d}(n)$. Данная линеаризация состоит из частных производных p выходов всей сети по W весам модели:

$$\mathbf{C}(n) = \begin{bmatrix} \frac{\partial c_1}{\partial w_1} & \frac{\partial c_1}{\partial w_2} & \cdots & \frac{\partial c_1}{\partial w_W} \\ \frac{\partial c_2}{\partial w_1} & \frac{\partial c_2}{\partial w_2} & \cdots & \frac{\partial c_2}{\partial w_W} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial c_p}{\partial w_1} & \frac{\partial c_p}{\partial w_2} & \cdots & \frac{\partial c_p}{\partial w_W} \end{bmatrix}, \tag{15.72}$$

где $c_i, i = 1, 2, \ldots, p$, обозначает i-й элемент нелинейности $\mathbf{c}(\mathbf{w}(n), \mathbf{u}(n), \mathbf{v}(n))$. Частные производные в выражении (15.72) оцениваются в точке $\mathbf{w}(n) = \hat{\mathbf{w}}(n)$, где $\hat{\mathbf{w}}(n)$ — оценка вектора весов $\mathbf{w}(n)$, вычисленная расширенным фильтром Калмана в момент времени n, при заданных данных наблюдений вплоть до момента времени $n-1$ [434]. На практике эти частные производные вычисляются с использованием алгоритма обратного распространения во времени или алгоритма рекуррентного обучения в реальном времени. В результате расширенный фильтр Калмана строится на одном из алгоритмов, описанных в разделах 15.7 и 15.8. Из этого следует, что $\mathbf{c}$ должна быть функцией действия рекуррентного узла, что уже и отмечалось выше. На самом деле для однослойной рекуррентной сети матрица $\mathbf{C}(n)$ может быть составлена из элементов матриц $\mathbf{\Lambda}_j(n)$, вычисленных алгоритмом RTRL (15.52). Таким образом, матрица измерения $\mathbf{C}(n)$ является динамической матрицей производных выходов сети по отношению к свободным параметрам. Так же как активность рекуррентного узла сети в момент времени $(n+1)$ является функцией от соответствующих значений на предыдущем шаге n, так и производная от значения активности рекуррентного узла по отношению к свободным параметрам сети в момент времени $n+1$ является функцией от соответствующих значений на предыдущем шаге n (что следует из уравнения RTRL).

Теперь предположим, что синаптические веса сети разбиты на g групп и каждая группа i содержит k_i нейронов. Матрица измерения $\mathbf{C}$, определяемая формулой (15.72), является матрицей размерности $p \times W$, в которой содержатся производные от выхода сети по всем весам сети. Зависимость матрицы $\mathbf{C}(n)$ от входного вектора $\mathbf{u}(n)$ в формуле (15.72) присутствует неявно. Таким образом, определенная матрица $\mathbf{C}(n)$ содержит все производные, которые необходимы для любой несвязной версии расширенного фильтра Калмана. Например, если используется *глобальный расширен-*

ный фильтр Калмана (global extended Kalman filter — GEKF), т.е. отсутствует несвязность, то $g = 1$ и вся матрица $\mathbf{C}(n)$ определяется выражением (15.72). С другой стороны, если используется *несвязный расширенный фильтр Калмана* (decoupled extended Kalman filter — DEKF), то "глобальная" матрица измерения $\mathbf{C}(n)$ должна быть скомпонована таким образом, чтобы веса, соответствующие конкретным нейронам в сети, группировались в единый блок внутри матрицы, и каждый блок маркировался соответствующим индексом $i = 1, 2, \ldots, g$. В последнем случае общая матрица $\mathbf{C}(n)$ является обычным *объединением* матриц отдельных групп $\mathbf{C}_i(n)$:

$$\mathbf{C}(n) = [\mathbf{C}_1(n), \mathbf{C}_2(n), \ldots, \mathbf{C}_g(n)].$$

В любом случае, независимо от уровня несвязности, вся матрица $\mathbf{C}(n)$ должна вычисляться так, как определено в формуле (15.72).

Итак, мы подготовили почву для применения алгоритма фильтрации Калмана (см. табл. 15.2). В частности, для линеаризованной динамической модели, описываемой уравнениями (15.69) и (15.70), получим [864]:

$$\mathbf{\Gamma}(n) = \left[\sum_{i=1}^{g} \mathbf{C}_i(n)\mathbf{K}_i(n, n-1)\mathbf{C}_i^T(n) + \mathbf{R}(n)\right]^{-1}, \tag{15.73}$$

$$\mathbf{G}_i(n) = \mathbf{K}_i(n, n-1)\mathbf{C}_i^T(n)\mathbf{\Gamma}(n), \tag{15.74}$$

$$\boldsymbol{\alpha}(n) = \mathbf{d}(n) - \hat{\mathbf{d}}(n|n-1), \tag{15.75}$$

$$\hat{\mathbf{w}}_i(n+1|n) = \hat{\mathbf{w}}_i(n|n-1) + \mathbf{G}_i(n)\boldsymbol{\alpha}(n), \tag{15.76}$$

$$\mathbf{K}_i(n+1, n) = \mathbf{K}_i(n, n-1) - \mathbf{G}_i(n)\mathbf{C}_i(n)\mathbf{K}_i(n, n-1), \tag{15.77}$$

где $i = 1, 2, \ldots, g$. Вектор параметров и векторы сигналов в формулах (15.73)–(15.77) описываются следующим образом.

$\mathbf{\Gamma}(n)$ — матрица размерности $p \times p$, представляющая собой глобальный коэффициент передачи для всей сети.

$\mathbf{G}_i(n)$ — матрица размерности $W_i \times p$, обозначающая коэффициент усиления Калмана для группы i нейронов.

$\boldsymbol{\alpha}(n)$ — вектор размерности $p \times 1$, содержащий инновации, определяемые как разность между желаемым откликом $\mathbf{d}(n)$ линеаризованной системы и его оценкой $\hat{\mathbf{d}}(n|n-1)$, основанной на входных данных, доступных в момент времени $n-1$. Эта оценка представлена вектором фактического выхода $\mathbf{y}(n)$ сети, находящейся в состоянии $\{\hat{\mathbf{w}}_i(n|n-1)\}$, который является откликом сети на подачу входного сигнала $\mathbf{u}(n)$.

$\hat{\mathbf{w}}_i(n|n-1)$ — вектор размерности $W \times 1$, являющийся оценкой вектора весов $\mathbf{w}_i(n)$ для группы i в момент времени n, при наличии наблюдаемых данных вплоть до момента времени $n-1$.

$\mathbf{K}_i(n, n-1)$ — матрица размерности $k_i \times k_i$, являющаяся матрицей ковариации ошибок группы нейронов i.

Суммирование, выполняемое при вычислении глобального коэффициента передачи $\Gamma(n)$ (15.73), учитывает несвязную природу расширенного фильтра Калмана.

Важно понять, что в алгоритме DEKF несвязность определяет, какие конкретные элементы глобальной матрицы ковариации ошибок $\mathbf{K}(n, n-1)$ должны рассматриваться и корректироваться. На самом деле вся экономия вычислительных ресурсов связана только с игнорированием операций по хранению и корректировке ассоциированных с отстоящими от диагонали блоками глобальной матрицы ковариации ошибок $\mathbf{K}(n, n-1)$. В противном случае пришлось бы объединять все группы синаптических весов.

Алгоритм DEKF, представленный формулами (15.73)–(15.77), минимизирует функцию стоимости

$$\mathbf{E}(n) = \frac{1}{2}\sum_{j=1}^{n} \|\mathbf{e}(j)\|^2, \tag{15.78}$$

где $\mathbf{e}(j)$ — вектор ошибок, определенный следующим образом:

$$\mathbf{e}(j) = \mathbf{d}(n) - \mathbf{y}(n), j = 1, 2, \ldots, n,$$

где $\mathbf{y}(j)$ — фактический выход сети, использующей информацию, доступную вплоть до момента j (включительно). Обратите внимание, что в общем случае $\mathbf{e}(j) \neq \alpha(j)$.

Искусственный шум процесса

Нейродинамическая система моделируется уравнениями (15.69) и (15.70), *не усилена* (unforced), так как уравнение процесса (15.69) не имеет внешних входных сигналов. Это может привести к серьезным вычислительным проблемам и, как следствие, к расходимости фильтра Калмана, когда он работает в среде с ограниченной точностью. Как уже говорилось в разделе 15.9, явление расходимости (дивергенции) можно обойти, использовав фильтрацию на основе квадратного корня.

Еще одним способом избежать расходимости является использование эвристического механизма, который предполагает искусственное добавление в уравнение процесса *шума процесса* (process noice):

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \boldsymbol{\omega}_i(n), i = 1, 2, \ldots, g, \tag{15.79}$$

где $\boldsymbol{\omega}_i(n)$ — шум процесса. Предполагается, что $\boldsymbol{\omega}_i(n)$ — многомерный белый шум с нулевым средним и матрицей ковариации $\mathbf{Q}_i(n)$. Искусственно добавленный шум процесса $\boldsymbol{\omega}_i(n)$ не зависит от шума измерения $\mathbf{v}_i(n)$ и начального состояния сети. Общий эффект от добавления $\boldsymbol{\omega}_i(n)$ в уравнение процесса (15.79) состоит в изменении уравнения Рикатти коррекции матрицы ковариации ошибок следующим образом [433]:

$$\mathbf{K}_i(n+1,n) = \mathbf{K}_i(n,n-1) - \mathbf{G}_i(n)\mathbf{C}_i(n)\mathbf{K}_i(n,n-1) + \mathbf{Q}_i(n). \tag{15.80}$$

Если матрица $\mathbf{Q}_i(n)$ является достаточно большой для всех i, то $\mathbf{K}_i(n+1,n)$ будет гарантировано оставаться неотрицательно определенной для всех n.

Кроме исключения указанных выше вычислительных сложностей, искусственное добавление шума процесса $\boldsymbol{\omega}_i(n)$ в уравнение процесса дает еще один выигрышный эффект: в процессе обучения менее вероятно попадание алгоритма в точку локального минимума. Это, в свою очередь, приводит к существенному улучшению производительности обучения в терминах скорости сходимости и качества решения [864].

Полное описание алгоритма DEKF

В табл. 15.3 представлено полное описание алгоритма DEKF, основанного на формулах (15.73)–(15.76) и (15.80). В эту таблицу также включены подробности инициализации этого алгоритма.

Нам остается только привести некоторые заключительные комментарии относительно расширенного фильтра Калмана. Алгоритм DEKF (см. табл. 15.3) относится к семейству всех возможных *процедур обучения с сохранением информации* (information-preserving learning procedure), к числу которых принадлежит и алгоритм GEKF. Как правило, ожидается, что DEKF достигнет производительности алгоритма GEKF, но не сможет превзойти ее. С другой стороны, алгоритм DEKF менее требователен к вычислительным мощностям, чем GEKF. Вопреки этому вычислительному преимуществу, современные вычислительные ресурсы и объемы памяти сделали алгоритм GEKF доступным для некоторых практических задач, особенно в области *пакетного обучения* (off-line training) рекуррентных сетей.

Вычислительная сложность

В табл. 15.4 сравнивается вычислительная сложность трех алгоритмов обучения, описанных в этой главе: обратного распространения во времени, рекуррентного обучения в реальном времени и несвязной расширенной фильтрации Калмана.

15.11. Компьютерное моделирование

В этом эксперименте мы вернемся к моделированию нелинейных временных рядов, которые рассматривались в разделе 13.5. Этот временной ряд определяется частотно-модулированным сигналом:

$$x(n) = \sin(n + \sin(n^2)), n = 0, 1, 2, \ldots.$$

Для моделирования будем использовать две различные структуры.

ТАБЛИЦА 15.3. Полный алгоритм DEKF

Инициализация

1. Присваиваем синаптическим весам рекуррентной сети малые значение, выбираемые из некоторого равномерного распределения.
2. Присваиваем диагональным элементам матрицы ковариации $\mathbf{Q}(n)$ (характеризующей искусственно добавляемый шум процесса $\omega(n)$) значения из интервала $[10^{-6}, 10^{-2}]$.
3. Присваиваем $\mathbf{K}(0, 1)=\delta^{-1}\mathbf{I}$, где δ — некоторое малое положительное число.

Вычисления

$$\boldsymbol{\Gamma}(n) = \left[\sum_{i=1}^{g} \mathbf{C}_i(n)\mathbf{K}_i(n, n-1)\mathbf{C}_i^T(n) + \mathbf{R}(n)\right]^{-1},$$

$$\mathbf{G}_i(n) = \mathbf{K}_i(n, n-1)\mathbf{C}_i^T(n)\boldsymbol{\Gamma}(n),$$

$$\boldsymbol{\alpha}(n) = \mathbf{d}(n) - \hat{\mathbf{d}}(n|n-1),$$

$$\hat{\mathbf{w}}_i(n+1|n) = \hat{\mathbf{w}}_i(n|n-1) + \mathbf{G}_i(n)\boldsymbol{\alpha}(n),$$

$$\mathbf{K}_i(n+1, n) = \mathbf{K}_i(n, n-1) - \mathbf{G}_i(n)\mathbf{C}_i(n)\mathbf{K}_i(n, n-1) + \mathbf{Q}_i(n),$$

где в третьей строке $\hat{\mathbf{d}}(n|n-1)$ — реальный выходной вектор сети $\mathbf{y}(n)$, сгенерированный в ответ на входной вектор $\mathbf{u}(n)$

Примечание. При $g = 1$ (т.е. при отсутствии несвязности) алгоритм DEKF становится глобальным алгоритмом расширенной фильтрации Калмана (GEKF).

- *Рекуррентный многослойный персептрон* (recurrent multilayer perceptron — RMLP), который состоит из одного входного узла, первого скрытого слоя с десятью рекуррентными нейронами, второго скрытого слоя с десятью нейронами и одного линейного выходного нейрона.
- *Сеть прямого распространения со сфокусированной задержкой во времени* (time lagged feedforward network — TLFN), которая состоит из памяти с задержкой по времени с двадцатью отводами и многослойного персептрона с десятью скрытыми и одним линейным выходным нейроном.

Персептрон RMLP имеет чуть больше синаптических весов, чем фокусированная TLFN, однако для его хранения требуется в 2 раза меньше памяти (10 рекуррентных узлов против 20 элементов).

Персептрон RMLP обучается с помощью алгоритма DEKF. Сеть TLFN обучается с использованием двух версий расширенного фильтра Калмана: алгоритма GEKF (т.е. глобальной версии) и алгоритма DEKF (т.е. его несвязной версии). Характеристики этих двух алгоритмов приведены ниже.

ТАБЛИЦА 15.4. Сравнение вычислительной мощности различных алгоритмов обучения рекуррентных сетей

S — количество состояний
W — количество синаптических весов
L — длина последовательности обучения
1. Алгоритм BPTT (обратного распространения во времени):
Время $O(WL+SL)$
Требования к памяти $O(WL+SL)$
2. Алгоритм RTRL (рекуррентного обучения в реальном времени):
Время $O(WS^2L)$
Требования к памяти $O(WS)$
3. Алгоритм DEKF (несвязный расширенной фильтрации Калмана):
Алгоритм DEKF одинаково затратен (по времени и пространству) при вычислении производных алгоритмами RTRL и BPTT. При использовании последнего требования ко времени и к объему памяти умножаются на p, где p — количество выходов сети, по сравнению со стандартными затратами BPTT, в котором вычисляется одно слагаемой скалярной ошибки.
Алгоритм DEKF требует времени $O(p^2W + p\sum_{i=1}^{g} k_i^2)$ и объем памяти $O(\sum_{i=1}^{g} k_i^2)$, где g — количество групп; k_i — количество нейронов в группе i. В предельном случае, когда $g = 1$ (т.е. существует одна группа) и алгоритм сводится к GEKF, время составит $O(pW^2)$, а объем памяти — $O(W^2)$

- GEKF

  δ — параметр, используемый для инициализации матрицы ковариации ошибок $\mathbf{K}(n, n-1)$, равен 0,01.

  $\mathbf{R}(n)$ — матрица ковариации погрешности измерения $\mathbf{v}(n)$: $\mathbf{R}(0)$=100 в начале обучения, а затем постепенно снижается до $\mathbf{R}(n)$ =3 в конце обучения.

  $\mathbf{Q}(n)$ — матрица ковариации искусственного шума процесса $\omega(n)$: $\mathbf{Q}(0)$=10^{-2} в начале обучения, затем постепенно снижается до $\mathbf{Q}(n) = 10^{-6}$ в конце обучения.

  Отжиг матриц $\mathbf{R}(n)$ и $\mathbf{Q}(n)$ имеет эффект увеличения параметра скорости обучения по мере продвижения обучения.

- DEKF

  g — количество групп равно 21 для RMLP и 11 — для фокусированного TLFN.

  Все остальные параметры такие же, которые используются для GEKF.

Обучение осуществляется на последовательности, состоящей из 4000 образов. Для RMLP используются подмножества, состоящие из 100 примеров, при этом в ходе процесса обучения обрабатывается 30000 таких подмножеств. Каждая точка данных из множества 4000 примеров обрабатывается примерно 750 раз. В фокусированном

TLFN каждая точка множества обучения тоже обрабатывается около 750 раз. В обоих случаях тестирование осуществляется на 300 точках данных.

На рис. 15.14 показан график одношагового прогнозирования $\hat{y}(n)$, вычисленного персептроном RMLP, обученным алгоритмом DEKF. На этом же рисунке показан график фактического сигнала $y(n)$. Эти два графика трудно отличить друг от друга.

На рис. 15.15, *а* показана ошибка прогнозирования

$$e(n) = y(n) - \hat{y}(n)$$

алгоритма RMLP. Соответствующие ошибки прогнозирования, полученные фокусированной сетью TLFN, обученной алгоритмами GEKF и DEKF, показаны на рис. 15.15, *б* и *в*. Сравнивая полученные результаты между собой и с результатами, полученными в разделе 13.5, можно сделать следующие выводы.

1. Наиболее точное моделирование в смысле среднеквадратической ошибки было получено персептроном RMLP, обученным алгоритмом DEKF. Дисперсия ошибки прогнозирования, вычисленная на 5980 примерах, составила $1,1839 \times 10^{-4}$.
2. Для фокусированной TLFN наиболее точное моделирование в смысле среднеквадратической ошибки было получено при обучении алгоритмом GEKF. При использовании этого обучения дисперсия ошибки прогнозирования составила $1,3351 \times 10^{-4}$, в то время как при обучении DEKF — $1,5871 \times 10^{-4}$. Оба результата получены с использованием 5980 примеров.
3. Для сети TLFN, обучаемой стандартным алгоритмом обратного распространения, дисперсия ошибки прогнозирования составила $1,2 \times 10^{-3}$ (см. раздел 13.5). Эта величина на порядок хуже показателей, полученных с помощью алгоритмов GEKF и DEKF.

Превосходство в производительности расширенного алгоритма фильтрации Калмана над алгоритмом обратного распространения получено благодаря свойству *сохранения информации* (information-preserving) первого.

15.12. Обращение в нуль градиентов в рекуррентных сетях

При практическом применении рекуррентных сетей особого внимания может потребовать *проблема обращения градиентов в нуль* (vanishing gradients problem), которая связана с обучением сетей желаемому отклику в настоящем, который зависит от входных данных в далеком прошлом [121], [469]. Эта проблема возникает по следующей причине: комбинированные нелинейности приводят к тому, что бесконечно малые

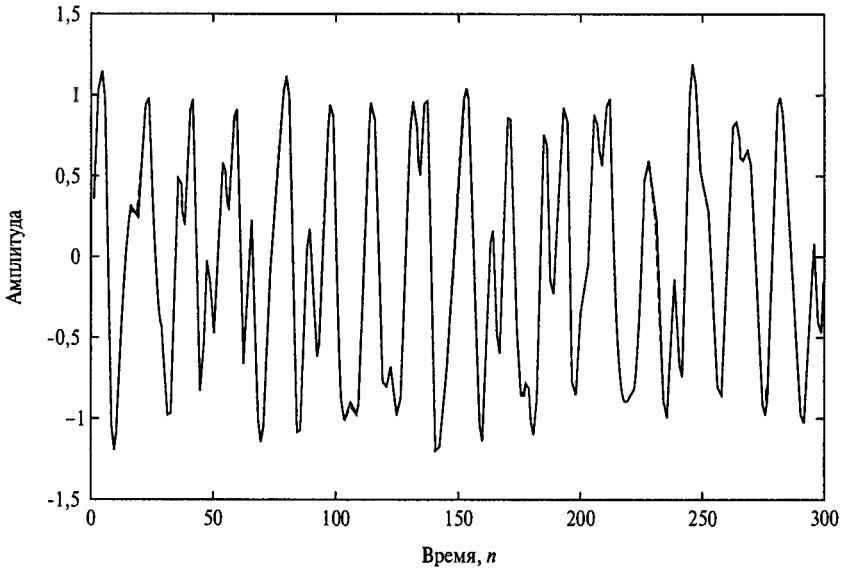


Рис. 15.14. Наложение графиков фактического (сплошная линия) и прогнозируемого (пунктирная линия) сигналов в данном компьютерном моделировании. Прогнозирование осуществлялось персептроном RMLP, обученным алгоритмом DEKF

изменения в удаленном во времени входном сигнале могут практически не оказывать воздействие на обучение сети. Эта проблема может возникнуть и в случае, когда большие изменения в отдаленных во времени входных сигналах оказывают влияние, но этот эффект не измерим градиентом. Описанная проблема делает обучение долгосрочным зависимостям в градиентных алгоритмах обучения чрезвычайно сложным, а в некоторых случаях даже виртуально невозможным.

В [121] было доказано, что во многих практических приложениях необходимо, чтобы сеть была способна хранить информацию о своем состоянии для произвольных отрезков времени, при этом делать это в присутствии шума. Долгосрочное хранение определенных битов информации в переменных состояния рекуррентной сети называется *блокированием информации* (information latching). Блокирование информации должно быть *робастным* (robust), чтобы сохраненная информация о состоянии не могла быть легко удалена событиями, которые не связаны с текущей задачей обучения. В этих терминах можно сделать следующее утверждение [121].

Робастное блокирование информации в рекуррентной сети выполнимо, если состояния сети содержатся в сокращенном множестве аттракции гиперболического аттрактора.

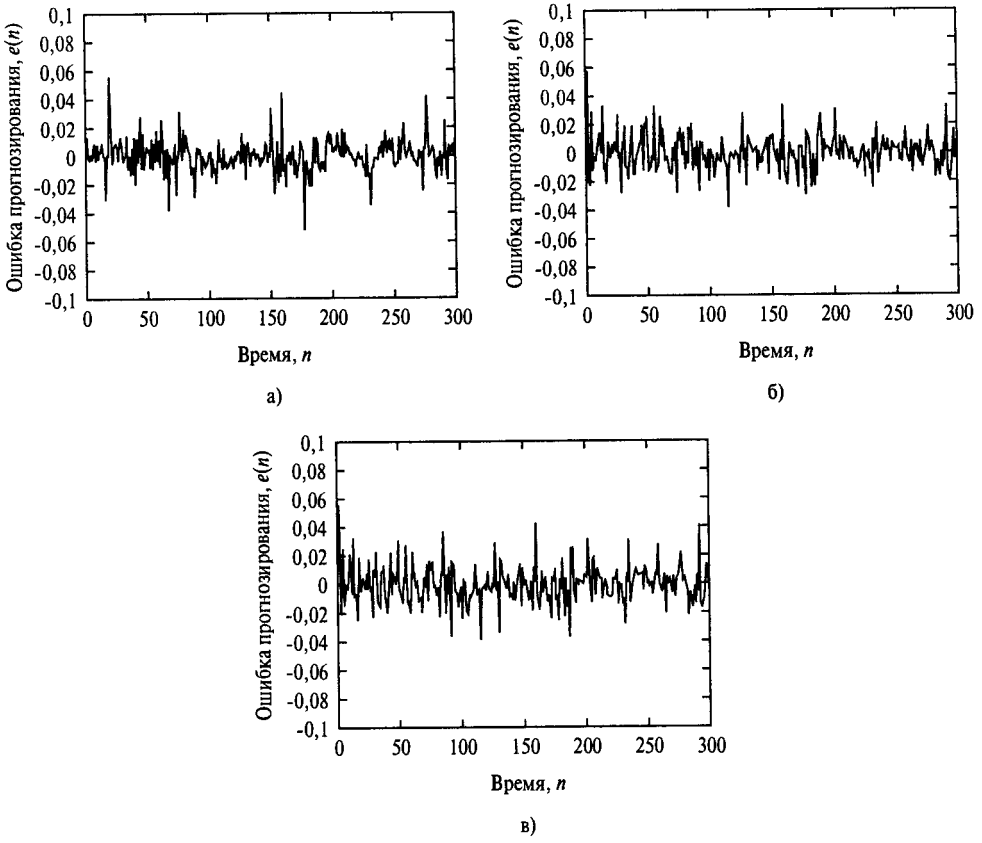


Рис. 15.15. Ошибка прогнозирования в трех экспериментах по моделированию: с помощью RMLP, обученного алгоритмом DEKF, — дисперсия ошибки составила $1,1839 \times 10^{-4}$ (а); с помощью фокусированной версии TLFN, обученной алгоритмом GEKF, — дисперсия ошибки составила $1,3351 \times 10^{-4}$ (б); с помощью фокусированной TLFN, обученной алгоритмом DEKF, — дисперсия ошибки составила $1,5871 \times 10^{-4}$ (в)

Понятие гиперболического аттрактора было введено в главе 14. *Сокращенным множеством аттракции* (reduced attraction set) гиперболического аттрактора является множество точек в бассейне аттракции, для которых собственные значения соответствующего Якобиана меньше единицы по модулю. Смысл здесь заключается в том, что если состояние $\mathbf{x}(n)$ рекуррентной сети находится в бассейне аттракции гиперболического аттрактора, но не принадлежит сокращенному множеству аттракции, то размер шара неопределенности вокруг $\mathbf{x}(n)$ будет расти экспоненциально времени n (см. рис. 15.16, *а*). Таким образом, малые возмущения (например, помехи) во входном сигнале рекуррентной сети могут вывести траекторию в другой (возможно, неверный) бассейн аттракции. Если же состояние $\mathbf{x}(n)$ остается в сокращенном множестве аттракции гиперболического аттрактора, можно найти ограничения входного сигнала, которые будут гарантировать нахождение $\mathbf{x}(n)$ в пределах заданной окрестности аттрактора (см. рис. 15.16, *б*).

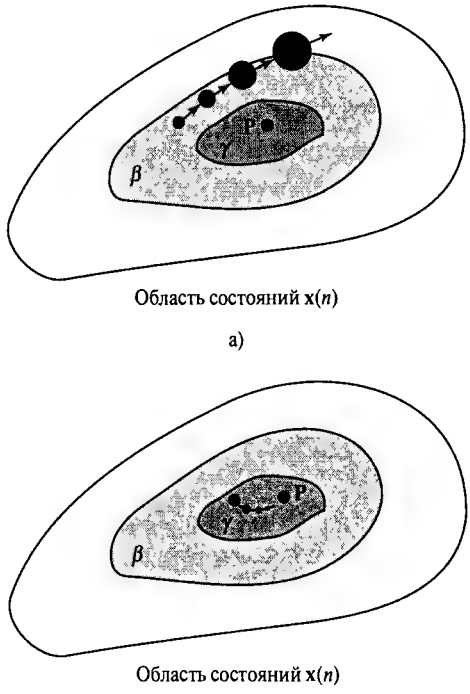


Рис. 15.16. Состояние $\mathbf{x}(n)$ находится в бассейне аттракции β, но вне сокращенного множества аттракции γ (а); состояние $\mathbf{x}(n)$ находится внутри сокращенного множества аттракции γ (б)

Долгосрочные зависимости

Чтобы оценить влияние робастного блокирования информации в градиентном обучении, обратим внимание, что коррекция, применяемая к вектору весов $\mathbf{w}$ рекуррентной сети в момент времени n, определяется формулой

$$\Delta\mathbf{w}(n) = -\eta\frac{\partial\mathbf{E}_{\text{общ.}}}{\partial\mathbf{w}},$$

где η — параметр скорости обучения; $\partial\mathbf{E}_{\text{общ.}}/\partial\mathbf{w}$ — градиент функции стоимости $\mathbf{E}_{\text{общ.}}$ по $\mathbf{w}$. Сама функция стоимости обычно определяется следующей формулой:

$$\mathbf{E}_{\text{общ.}} = \frac{1}{2}\sum_i \|\mathbf{d}_i(n) - \mathbf{y}_i(n)\|^2,$$

где $\mathbf{d}_i(n)$ — желаемый отклик; $\mathbf{y}_i(n)$ — фактический отклик сети на подачу i-го примера в момент времени n. Исходя из этого, можно записать:

$$\Delta \mathbf{w}(n) = -\eta \sum_i \frac{\partial \mathbf{y}_i(n)}{\partial \mathbf{w}} (\mathbf{d}_i(n) - \mathbf{y}_i(n)) \tag{15.81}$$
$$= \eta \sum_i \frac{\partial \mathbf{y}_i(n)}{\partial \mathbf{x}_i(n)} \frac{\partial \mathbf{x}_i(n)}{\partial \mathbf{w}} (\mathbf{d}_i(n) - \mathbf{y}_i(n)),$$

где во второй строке использовалось *цепное правило вычислений* (chain rule of calculus). Вектор состояния $\mathbf{x}_i(n)$ соответствует i-му примеру множества обучения. При использовании алгоритмов, в том числе обратного распространения во времени, частные производные функции стоимости вычисляются по независимым весам с различными индексами времени. Результат (15.81) можно несколько расширить, записав:

$$\Delta \mathbf{w}(n) = \eta \sum_i \frac{\partial \mathbf{y}_i(n)}{\partial \mathbf{x}_i(n)} \sum_{k=1}^{n} \frac{\partial \mathbf{x}_i(n)}{\partial \mathbf{w}(k)} (\mathbf{d}_i(n) - \mathbf{y}_i(n)).$$

Применяя цепное правило вычислений еще раз, получим:

$$\Delta \mathbf{w}(n) = \eta \sum_i \frac{\partial \mathbf{y}_i(n)}{\partial \mathbf{x}_i(n)} \sum_{k=1}^{n} \frac{\partial \mathbf{x}_i(n)}{\partial \mathbf{x}_i(k)} \frac{\partial \mathbf{x}_i(k)}{\partial \mathbf{w}(k)} (\mathbf{d}_i(n) - \mathbf{y}_i(n)). \tag{15.82}$$

Учитывая уравнение состояния (15.2), имеем:

$$\mathbf{x}_i(n) = \boldsymbol{\varphi}(\mathbf{x}_i(k), \mathbf{u}(n)), 1 \leq k < n.$$

Исходя из этого, выражение $\partial \mathbf{x}_i(n)/\partial \mathbf{x}_i(k)$ можно интерпретировать как Якобиан нелинейной функции $\boldsymbol{\varphi}(\cdot, \cdot)$, развернутый на $(n-k)$ шагов времени, т.е.

$$\frac{\partial \mathbf{x}_i(n)}{\partial \mathbf{x}_i(k)} = \frac{\partial \boldsymbol{\varphi}(\mathbf{x}_i(k), \mathbf{u}(n))}{\partial \mathbf{x}_i(k)} = \mathbf{J}_{\mathbf{x}}(n, n-k). \tag{15.83}$$

В [121] было показано следующее. Если входной сигнал $\mathbf{u}(n)$ является таковым, что рекуррентная сеть остается робастно заблокированной в гиперболическом аттракторе после момента времени $n = 0$, то Якобиан $\mathbf{J}_{\mathbf{x}}(n, k)$ является экспоненциально убывающей функцией k и

$$\det(\mathbf{J}_{\mathbf{x}}(n, k)) \to 0 \text{ при } k \to \infty \text{ для всех } n. \tag{15.84}$$

Значение результата (15.84) состоит в том, что *малое* изменение вектора весов $\mathbf{w}$ сети ощущается в основном в ближайшем прошлом (т.е. значения k близки к текущему времени n). Может существовать такое приращение $\Delta \mathbf{w}$ к вектору весов $\mathbf{w}$ в момент времени n, которое позволит переместиться из текущего состояния $\mathbf{x}(n)$ в другой, возможно лучший, бассейн аттракции, однако градиент функции стоимости $\mathbf{E}_{\text{общ.}}$ по $\mathbf{w}$ не учитывает эту информацию.

Если для хранения информации о состоянии в рекуррентной сети, обучаемой градиентным методом, используются гиперболические аттракторы, оказывается, что

- либо сеть *не робастна* к наличию шума во входном сигнале,
- либо сеть не способна извлечь *долгосрочные зависимости* (long-term dependencies) (т.е. связи целевых выходов с входными сигналами из далекого прошлого).

Среди возможных процедур облегчения сложностей, возникающих в связи с обращением в нуль градиентов в рекуррентных сетях, можно отметить следующие[13].

- Расширение временного диапазона зависимости выхода от входа с помощью представления сети во время обучения сначала более коротких строк (см. эвристики в разделе 15.6).
- Использование расширенного фильтра Калмана или ее несвязной версии для более эффективного использования доступной информации, чем в алгоритмах градиентного обучения (расширенный фильтр Калмана рассматривался в разделе 15.10).
- Использование более сложных методов оптимизации, таких как псевдоньютоновские методы [121]; методов оптимизации второго порядка или моделирование отжига (см. главы 4 и 11).

15.13. Системная идентификация

Идентификация систем (system identification) — это экспериментальный подход к моделированию процессов или производств (plant) с неизвестными параметрами[14]. Она включает в себя следующие действия: планирование эксперимента, выбор структуры модели, оценку параметров и проверку модели на корректность. Процедура идентификации системы, как показала практика, является итеративной по своей природе.

Предположим, существует некоторый неизвестный динамический объект, для которого требуется решить задачу параметрической идентификации модели. При этом можно основываться на модели в пространстве состояний или на модели в терминах "вход-выход". Решение относительно использования той или иной модели зависит от конкретной информации о входах и наблюдениях в системе. В следующем разделе описываются оба эти представления.

---

[13] Другие методы устранения проблем обращения в нуль градиента включают в себя шунтирование некоторых из нелинейностей рекуррентной сети для улучшения обучеиия долгосрочным зависимостям. Среди примеров этого подхода можно выделить следующие.

Использование в сетевой архитектуре долгосрочиых задержек [280], [353], [645].

Иерархическое структурирование сети на несколько уровней, ассоциированных с различными масштабами времени [280].

Использование шлюзовых элементов (gate unit) для обхода некоторых нелиинейностей [470].

[14] Идентификации систем посвящено очень много киит, например [664], [666]. Обзор этого направления с акцентом на иейронные сети содержится в [772], [999]. Впервые идентификация систем с помощью нейронных сетей подробно описана в [774].

Идентификация систем с использованием модели в пространстве состояний

Предположим, что рассматриваемый объект описывается моделью в пространстве состояний:

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{x}(n), \mathbf{u}(n)), \tag{15.85}$$

$$\mathbf{y}(n) = \mathbf{h}(\mathbf{x}(n)), \tag{15.86}$$

где $\mathbf{f}(\cdot,\cdot)$ и $\mathbf{h}(\cdot)$ — неизвестные нелинейные вектор-функции; уравнение (15.86) является обобщением уравнения (15.11). Для идентификации системы будем использовать две нейронные сети: одну для работы с уравнением процесса (15.85), а вторую — для работы с уравнением измерения (15.86) (рис. 15.17).

Очевидно, что состояние $\mathbf{x}(n)$ можно рассматривать как задержанную на один шаг версию $\mathbf{x}(n+1)$. Обозначим $\hat{\mathbf{x}}(n+1)$ оценку состояния $\mathbf{x}(n+1)$, полученную первой нейронной сетью (на рис. 15.17, *а* она обозначена цифрой I). Эта сеть работает с объединенным входом, состоящим из входного сигнала $\mathbf{u}(n)$ и состояния $\mathbf{x}(n)$, и дает на выходе $\hat{\mathbf{x}}(n+1)$. Оценка $\hat{\mathbf{x}}(n+1)$ вычитается из фактического состояния $\mathbf{x}(n+1)$, из чего получается вектор ошибки:

$$\mathbf{e}_{\mathrm{I}}(n+1) = \mathbf{x}(n+1) - \hat{\mathbf{x}}(n+1).$$

Здесь $\mathbf{x}(n+1)$ играет роль желаемого отклика. Предполагается, что доступно фактическое состояние $\mathbf{x}(n)$. Вектор ошибки $\mathbf{e}_I(n+1)$, в свою очередь, используется для коррекции синаптических весов нейронной сети I (см. рис. 15.17, *а*) с целью минимизации функции стоимости, основанной на векторе ошибки $\mathbf{e}_I(n+1)$ в некотором статистическом смысле.

Вторая нейронная сеть (обозначенная цифрой II на рис. 15.17, *б*) работает с фактическим состоянием $\mathbf{x}(n)$ неизвестного объекта и дает на выходе оценку $\hat{\mathbf{y}}(n)$ фактического выходного сигнала $\mathbf{y}(n)$. Эта оценка вычитается из $\mathbf{y}(n)$, в результате чего получается второй вектор ошибки:

$$\mathbf{e}_{\mathrm{II}}(n) = \mathbf{y}(n) - \hat{\mathbf{y}}(n),$$

где $\mathbf{y}(n)$ играет роль желаемого отклика. Вектор $\mathbf{e}_{\mathrm{II}}(n)$ используется затем для коррекции синаптических весов нейронной сети II с целью минимизации Евклидовой нормы вектора ошибки $\mathbf{e}_{\mathrm{II}}(n)$ в некотором статистическом смысле.

Две нейронные сети, показанные на рис. 15.17, работают синхронно и дают на выходе решение в пространстве состояний задачи системной идентификации [774]. Такая модель называется *последовательно-параллельной моделью идентификации* (series-parallel identification model) в признание того факта, что модель идентификации получает фактическое состояние неизвестной системы (а не фактическое состояние

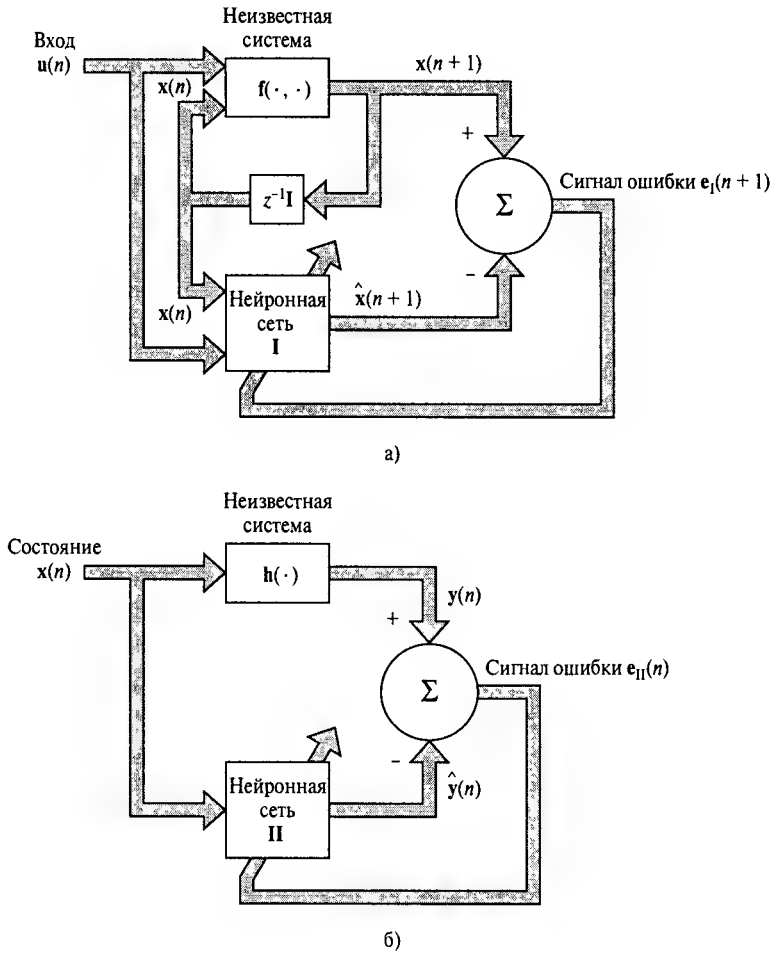


Рис. 15.17. Решение задачи идентификации систем с помощью модели в пространстве состояний

модели идентификации) (см. рис. 15.17, *а*). В свете обсуждения, представленного в конце раздела 15.9, эту форму обучения можно рассматривать как усиление учителем.

Последовательно-параллельная модель идентификации, показанная на рис. 15.17, *а*, противоположна *параллельной модели идентификации* (parallel identification model), в которой состояние $\mathbf{x}(n)$, подаваемое в нейронную сеть I, заменяется его оценкой $\hat{\mathbf{x}}(n)$. Эта оценка получается из собственного выхода нейронной сети $\hat{\mathbf{x}}(n+1)$ путем пропускания последнего через оператор единичной задержки $z^{-1}\mathbf{I}$. Практическое превосходство этой альтернативной модели обучения состоит в том, что нейросетевая модель работает в том же режиме, что и неизвестная система, т.е. в режиме, в котором сама нейронная сеть будет работать после обучения. Отсюда следует, что модель, полученная в результате параллельного обучения, скорее всего покажет более хорошую

динамику, чем модель сети, полученная в результате последовательно-параллельного обучения. Однако недостаток параллельной модели состоит в том, что ее обучение обычно продолжается дольше, чем последовательно-параллельной (см. раздел 15.9). В частности, в нашей конкретной ситуации оценка состояния $\hat{\mathbf{x}}(n)$, используемая в модели параллельного обучения, обычно не так точна, как фактическое состояние $\mathbf{x}(n)$, используемое в последовательно-параллельном режиме обучения.

Модель в терминах "вход-выход"

Далее предположим, что доступен только выход неизвестного объекта. Для того чтобы упростить представление, рассмотрим систему с одним входом и одним выходом. Пусть выход системы в дискретный момент времени n обозначается как $y(n)$, а ее вход — как $u(n)$. Тогда, выбирая для работы модель NARX, модель идентификации примет следующую форму:

$$\hat{y}(n+1) = \varphi(y(n), \ldots, y(n-q+1), u(n), \ldots, u(n-q+1)),$$

где q — порядок неизвестной системы. В момент времени $n+1$ известны q последних значений входного сигнала и q последних значений выходного. Выход модели $\hat{y}(n+1)$ представляет собой оценку фактического выхода $y(n+1)$. Оценка $\hat{y}(n+1)$ вычитается из $y(n+1)$, в результате чего получается сигнал ошибки:

$$e(n+1) = y(n+1) - \hat{y}(n+1).$$

Здесь величина $y(n+1)$ играет роль желаемого отклика. Ошибка $e(n+1)$ используется для коррекции синаптических весов нейронной сети с целью минимизации ошибки в некотором статистическом смысле. Модель идентификации, представленная на рис. 15.18, имеет последовательно-параллельную (т.е. усиленную учителем) форму, так как фактический выход системы (а не выход модели идентификации) замкнут на вход модели.

15.14. Адаптивное управление на основе эталонной модели

Еще одним важным приложением рекуррентных сетей является создание *системы управления с обратной связью* (feedback control system), в которой состояние объекта нелинейно объединяется с применяемым управлением [861], [863]. Структура этой системы существенно усложняется и другими факторами, такими как наличие неизмеряемых и случайных возмущений, возможность неуникальности операции обращения, а также присутствие ненаблюдаемых состояний объекта.

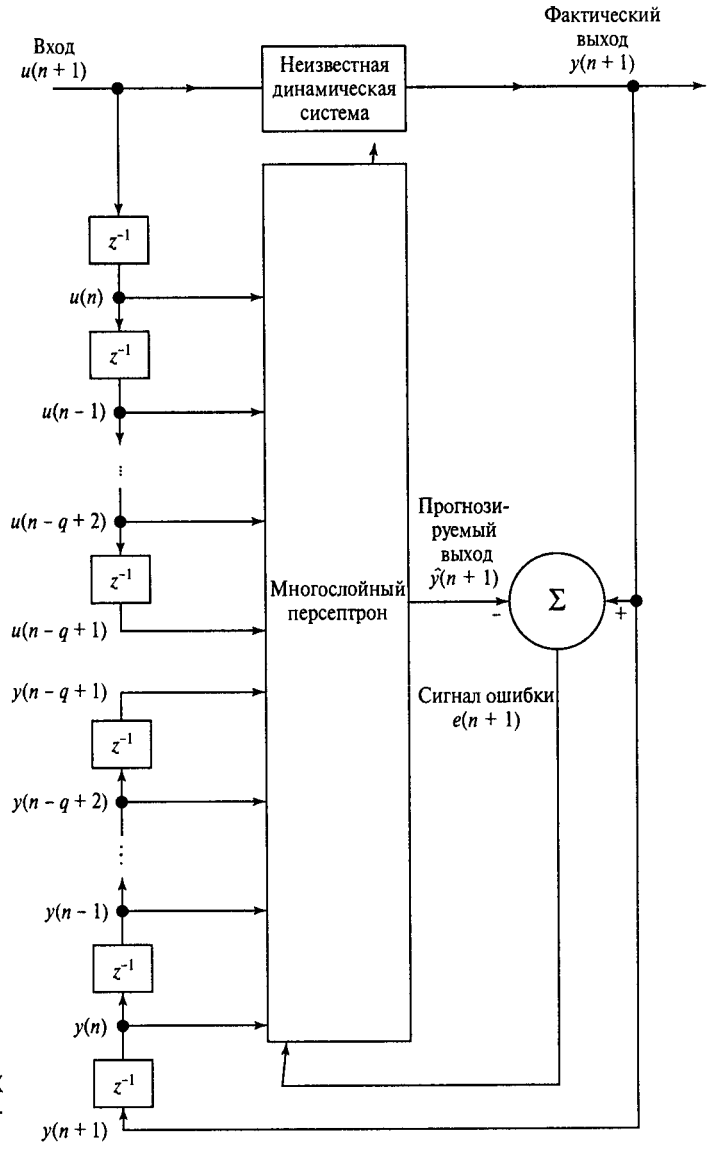


Рис. 15.18. Решение NARX задачи системной идентификации

Хорошо подходящей стратегией управления таким объектом с помощью нейронной сети является *адаптивное управление* на основе эталонной модели (model reference adaptive control — MRAC)[15]. В этой модели предполагается, что конструктор знаком с сущностью рассматриваемой системы [773]. На рис. 15.19 показана блочная диаграмма такой системы. Для учета того, что динамика объекта неизвестна, в ней используется адаптивность. Блок управления, или контроллер (controller), и объект

[15] Детальное описание адаптивного управления на основе модели содержится в [610].

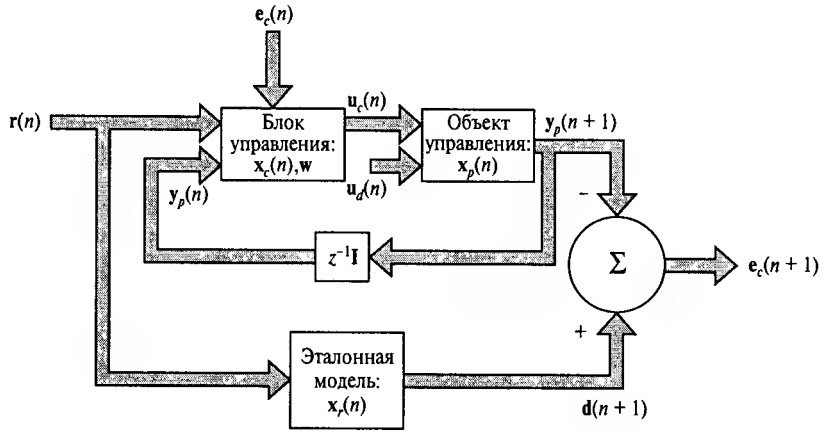


Рис. 15.19. Адаптивное управление на основе эталонной модели с использованием прямого управления

формируют систему с собственной обратной связью, образуя *внешне рекуррентную* (externally recurrent) сеть. Объект получает от блока управления входной сигнал $\mathbf{u}_c(n)$, а также внешнее возмущение $\mathbf{u}_d(n)$. Соответственно объект развивается во времени как функция описанного входного сигнала и собственного состояния $\mathbf{x}_p(n)$. Выход объекта, который обозначается как $\mathbf{y}_p(n+1)$, является функцией от $\mathbf{x}_p(n)$. Выход блока управления представляет собой вектор сигнала управления:

$$\mathbf{u}_c(n) = \mathbf{f}_1(\mathbf{x}_c(n), \mathbf{y}_p(n), \mathbf{r}(n), \mathbf{w}),$$

где $\mathbf{x}_c(n)$ — собственное состояние блока управления; $\mathbf{w}$ — вектор параметров, доступный для коррекции. Вектор-функция $\mathbf{f}_1(\cdot,\cdot,\cdot,\cdot)$ определяет динамику блока управления.

Желаемый отклик $\mathbf{d}(n+1)$ объекта формируется на выходе устойчивой *эталонной модели* (reference model), который образуется в ответ на *эталонный сигнал* (reference) $\mathbf{r}(n)$. Желаемый отклик $\mathbf{d}(n+1)$, таким образом, является функцией эталонного сигнала $\mathbf{r}(n)$ и собственного состояния $\mathbf{x}_r(n)$ эталонной модели:

$$\mathbf{d}(n+1) = \mathbf{f}_2(\mathbf{x}_r(n), \mathbf{r}(n)).$$

Вектор-функция $\mathbf{f}_2(\cdot,\cdot)$ определяет динамику эталонной модели.

Пусть *ошибка на выходе* (output error), т.е. разность между выходами объекта и эталонной модели, обозначается следующим образом:

$$\mathbf{e}_c(n+1) = \mathbf{d}(n+1) - \mathbf{y}_p(n+1).$$

Нашей целью является такая коррекция вектора параметров $\mathbf{w}$, чтобы Евклидова норма ошибки выхода $\mathbf{e}_c(n)$ была минимизирована для момента времени n.

Метод управления, используемый в MRAC-системе (см. рис. 15.19), назван *прямым* в том смысле, что для идентификации параметров объекта не предпринимается никаких действий, но при этом напрямую корректируются параметры блока управления для улучшения производительности системы. К сожалению, пока не существует точных методов настройки параметров блока управления, основанных на ошибке выхода [774], поскольку между блоком управления и ошибкой на выходе находится неизвестный объект. Чтобы обойти эту сложность, можно прибегнуть к *непрямому управлению* (indirect control) (рис. 15.20). В этом, последнем, методе для обучения блока управления используется двухшаговая процедура.

1. Для вывода оценок дифференциальных соотношений выхода объекта к его входу, предыдущего выхода объекта и предыдущих внутренних состояний объекта обсчитывается модель объекта P (которая обозначается $\hat{P}$). Для обучения нейронной сети идентификации объекта используется процедура, описанная в предыдущем разделе. Таким образом, полученная модель $\hat{P}$ называется *моделью идентификации* (identification model).
2. Для получения оценок динамических производных выхода объекта по отношению к вектору настраиваемых параметров блока управления вместо самого объекта используется полученная на предыдущем шаге модель идентификации $\hat{P}$.

При непрямом управлении *внешне рекуррентная сеть* (externally recurrent network) состоит из блока управления и модели идентификации $\hat{P}$, представляющей соотношение "вход-выход" объекта.

Применение рекуррентной сети к созданию блока управления в общей структуре рис. 15.20 было продемонстрировано на ряде примеров задач управления [861], [863]. В качестве рекуррентной сети, использованной в этом исследовании, выступает рекуррентный многослойный персептрон, аналогичный описанному в разделе 15.2. Обучение этой сети осуществлялось алгоритмом DEKF, описанным в разделе 15.11. Однако обратите внимание, что для управления холостым ходом двигателя в качестве модели идентификации выбиралась *линейная* динамическая система, так как оказалось, что применяемое управление (в соответствующим образом выбранных интервалах) монотонно влияет на скорость двигателя.

15.15. Резюме и обсуждение

В этой главе речь шла о рекуррентных сетях, которые используют *глобальную обратную связь* (global feedback), применяемую к статическому (не имеющему памяти) многослойному персептрону. Применение обратной связи позволяет нейронным се-

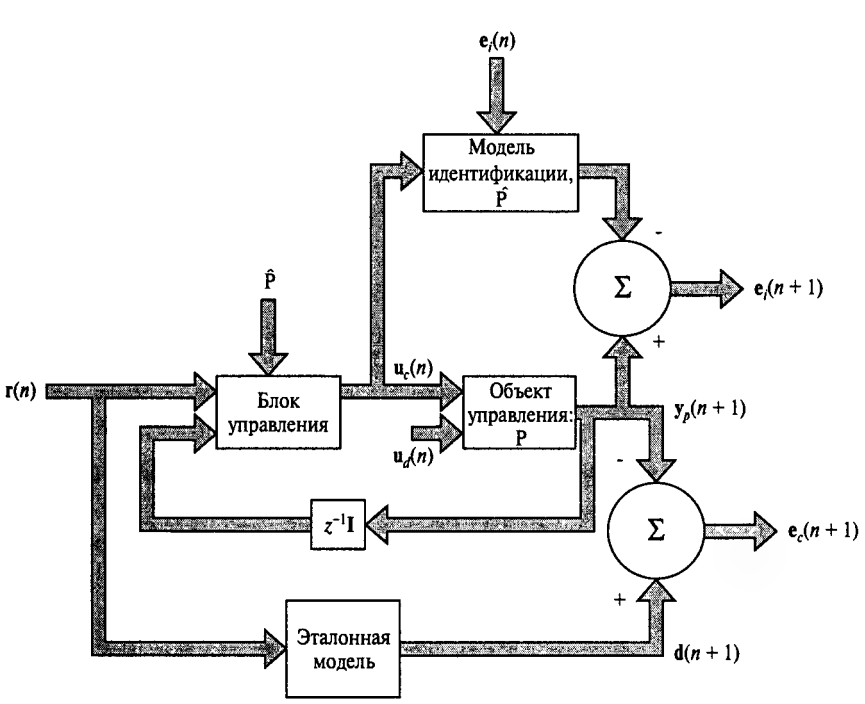


Рис. 15.20. Адаптивное управление на основе эталонной модели, использующее непрямое управление посредством модели идентификации

тям представлять состояния, что делает их удобным инструментом в различных приложениях обработки сигнала и управления. В класс рекуррентных сетей с глобальной обратной связью входят следующие основные архитектуры сетей.

- Сети нелинейной авторегрессии с внешними входами (NARX). В них используется обратная связь между выходным и входным слоями.
- Полносвязные рекуррентные сети с обратной связью между скрытым и входным слоем.
- Рекуррентный многослойный персептрон, содержащий больше одного скрытого слоя, с обратными связями между каждым из расчетных слоев и входным слоем.
- Рекуррентные сети второго порядка, использующие нейроны второго порядка.

Во всех этих рекуррентных сетях обратная связь применяется через память с дискретной задержкой в линии.

Первые три типа рекуррентных сетей допускают использование для изучения динамики *среду пространства состояний* (state-space framework). Этот подход берет начало в современной теории управления и является мощным методом изучения нелинейной динамики рекуррентных сетей.

В этой главе описаны три основных алгоритма обучения рекуррентных нейронных сетей: обратное распространение во времени (back-propagation through time — BPTT), рекуррентное обучение в реальном времени (real-time recurrent learning — RTRL) и несвязная расширенная фильтрация Калмана (decoupled extended Kalman filtering — DEKF). Первые два алгоритма основаны на градиентной методике, а последний эффективно использует информацию более высокого порядка. Это обеспечивает более быструю его сходимость по сравнению с алгоритмами BPTT и RTRL, однако за счет соответствующего повышения сложности вычислений. Алгоритм DEKF можно рассматривать как технологию *включения* (enabling), которая делает возможным решение сложных задач обработки сигнала и управления.

Теоретически рекуррентная сеть с глобальной обратной связью (например, рекуррентный многослойный персептрон, обучаемый алгоритмом DEKF) может обучиться *нестационарной* динамике за счет сохранения знаний, извлеченных из примеров обучения в *зафиксированном* множестве весов. И что более важно — эта сеть может *отслеживать* (track) изменения в статистике среды, если выполнены два условия.

- Данная сеть избавлена от недостаточного и чрезмерного обучения.
- Примеры обучения являются представительными в смысле нестационарности динамики среды.

В данной главе особое внимание уделялось использованию рекуррентных сетей для временной обработки. Рекуррентные сети можно также использовать для обработки последовательно упорядоченных данных, которые не имеют простой временной интерпретации (например, химические структуры, представленные в виде деревьев). В [1013] было показано, что рекуррентные сети могут представлять и классифицировать структурированные модели, которые представлены направленными, маркированными, ациклическими графами. Описанный в работе подход использует идею "обобщенного рекурсивного нейрона", являющегося обобщением обычного рекуррентного нейрона (т.е. нейрона с собственной обратной связью). Используя такую модель, алгоритмы обучения с учителем (такие как обратное распространение во времени и рекуррентное обучение в реальном времени) можно расширить для работы со структурированными *образами* (pattern).

Задачи

Модель в пространстве состояний

15.1. Сформулируйте уравнение в пространстве состояний для простой рекуррентной сети Элмана, показанной на рис. 15.3.

15.2. Покажите, что многослойный персептрон, показанный на рис. 15.4, может быть представлен следующей моделью в пространстве состояний:

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{x}(n), \mathbf{u}(n)),$$
$$\mathbf{y}(n) = \mathbf{g}(\mathbf{x}(n), \mathbf{u}(n)),$$

где $\mathbf{u}(n)$ — входной сигнал; $\mathbf{y}(n)$ — выходной сигнал; $\mathbf{x}(n)$ — состояние; $\mathbf{f}(\cdot,\cdot)$ и $\mathbf{g}(\cdot,\cdot)$ — нелинейные вектор-функции.

15.3. Могут ли динамические системы быть управляемыми, но не наблюдаемыми, и наоборот? Обоснуйте свой ответ.

15.4. Возвращаясь к задаче локальной управляемости (см. раздел 15.3), покажите, что:

а) состояние $\mathbf{x}(n+q)$ является вложенной нелинейной функцией от своего прошлого значения $\mathbf{x}(n)$ и входного вектора $\mathbf{u}_q(n)$ из (15.24) и

б) Якобиан $\mathbf{x}(n+q)$ по отношению к $\mathbf{u}_q(n)$, вычисленный в начале координат, равен матрице управляемости $\mathbf{M}_c$ из (15.23).

15.5. Возвращаясь к задаче локальной управляемости, рассмотренной в разделе 15.3, покажите, что Якобиан вектора наблюдения $\mathbf{y}_q(n)$, определяемый в (15.30), по отношению к состоянию $\mathbf{x}(n)$, вычисленный в начале координат, равен матрице наблюдаемости $\mathbf{M}_o$ из (15.28).

15.6. Уравнение процесса нелинейной динамической системы выглядит следующим образом:

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{x}(n), \mathbf{u}(n)),$$

где $\mathbf{u}(n)$ — вектор входа в момент времени n; $\mathbf{x}(n)$ — соответствующее состояние системы. Входной сигнал $\mathbf{u}(n)$ входит в уравнение процесса неаддитивно. В этой задаче необходимо переформулировать уравнение процесса так, чтобы входной сигнал входил в него аддитивно. Это можно сделать, записав:

$$\mathbf{x}'(n+1) = \mathbf{f}_{\text{new}}(\mathbf{x}'(n)) + \mathbf{u}'(n).$$

Определите векторы $\mathbf{u}'(n)$ и $\mathbf{x}'(n)$ и функцию $\mathbf{f}_{\text{new}}(\cdot)$ для этого уравнения.

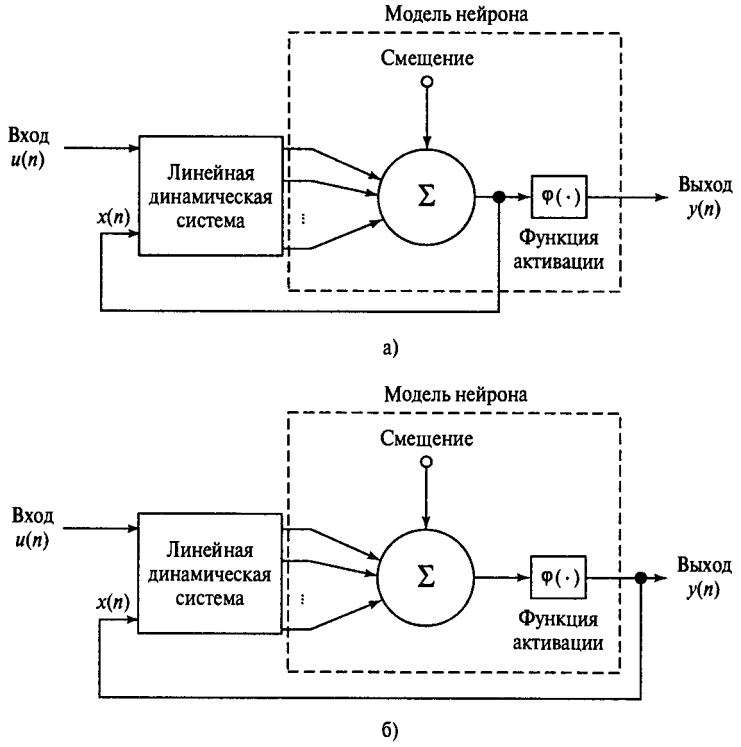


Рис. 15.21. Архитектура локальной обратной связи активации (а); архитектура локальной обратной связи выхода (б)

15.7. На рис. 15.21 показаны два примера архитектуры рекуррентных сетей, использующей локальную обратную связь на уровне нейрона. Архитектуры, показанные на рис. 15,21, *а*, *б*, называются соответственно *локальной обратной связью активации* (local activation feedback) и *локальной обратной связью выхода* (local output feedback) [1060]. Сформулируйте модели в пространстве состяний для этих архитектур рекуррентных сетей. Прокомментируйте их управляемость и наблюдаемость.

Модель нелинейной авторегрессии с экзогенными входами (NARX)

15.8. Возвращаясь к модели NARX, рассмотренной в разделе 15.4, покажите, что использование уравнений (15.16) и (15.17) ведет к следующей формуле для выхода $y(n+q)$ модели NARX в терминах векторов состояния $\mathbf{x}(n)$ и входа $\mathbf{u}_q(n)$:

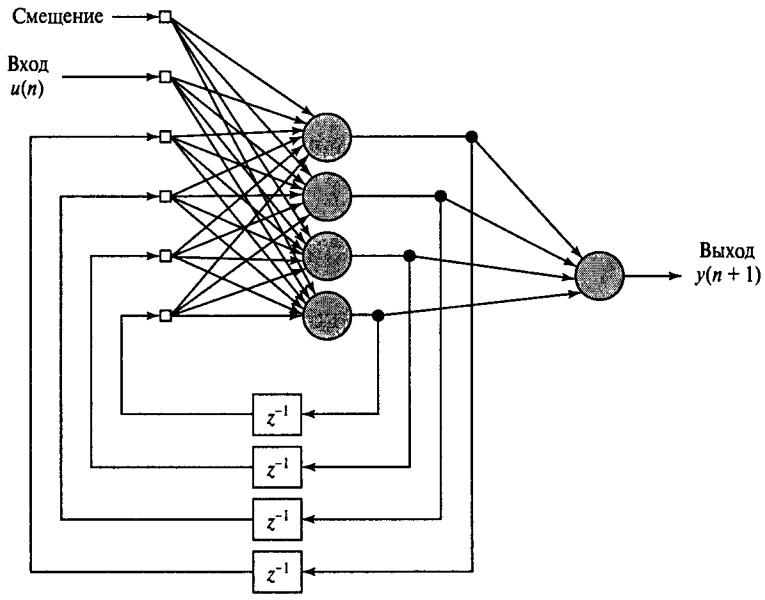


Рис. 15.22. Модель полностью рекуррентной сети

$$y(n+q) = \Phi(\mathbf{x}(n), \mathbf{u}_q(n)),$$

где $\Phi : \Re^{2q} \rightarrow \Re.$, а $\mathbf{u}_q(n)$ определяется согласно (15.29).

15.9. а) Вывод модели NARX в разделе 15.4 представлен для системы с одним входом и одним выходом. Обсудите, как описанная теория может быть расширена на систему с несколькими входами и выходами.

б) Постройте эквивалент NARX для модели в пространстве состояний системы с двумя входами и одним выходом (см. рис. 15.6).

15.10. Постройте эквивалентную сеть NARX для полностью рекуррентной сети, показанной на рис. 15.22.

15.11. В разделе 15.4 было показано, что модель в пространстве состояний может быть представлена моделью NARX. Можно ли утверждать обратное? Можно ли представить модель NARX в виде уравнений в пространстве состояний, описанной в разделе 15.3? Обоснуйте свой ответ.

Алгоритм BPTT

15.12. Раскройте временную динамику модели в пространстве состояний, показанной на рис. 15.3.

15.13. Усеченный алгоритм BPTT(h) можно рассматривать как аппроксимацию алгоритма обучения BPTT по эпохам. Эта аппроксимация может быть улучшена за счет введения аспектов алгоритма обучения по эпохам BPTT в усеченный алгоритм BPTT(h). В частности, можно заставить сеть пройти h' дополнительных шагов, прежде чем осуществлять следующее вычисление BPTT (где $h' < h$). Важной характеристикой этой гибридной формы обратного распространения во времени является то, что следующий обратный проход не будет выполняться до момента времени $n + h'$. В это время вмешательства прошлые значения входа сети, ее состояния и желаемых откликов сохраняются в буфере, но не обрабатываются [1155]. Выведите формулу локального градиента для нейрона j этого гибридного алгоритма.

Алгоритм рекуррентного обучения в реальном времени

15.14. Динамика *усиленной учителем рекуррентной сети* (teacher forced recurrent network) аналогична описанной в разделе 15.8, за исключением следующего изменения:

$$\xi_i(n) = \begin{cases} u_i(n), & \text{если } i \in \mathbf{A}, \\ d_i(n), & \text{если } i \in \mathbf{C}, \\ y_i(n), & \text{если } i \in \mathbf{B} - \mathbf{C}, \end{cases}$$

где $\mathbf{A}$ — множество индексов, для которых ξ_i является внешним входом; $\mathbf{B}$ — множество индексов, для которых ξ_i является выходом нейрона; $\mathbf{C}$ — множество видимых выходных нейронов.

а) Покажите, что для этой схемы частная производная $\partial y_i(n+1)/\partial w_{kl}(n)$ задается следующей формулой [1157]:

$$\frac{\partial y_j(n+1)}{\partial w_{kl}(n)} = \varphi'(v_j(n)) \left[\sum_{i \in \mathbf{B}-\mathbf{C}} w_{ji}(n) \frac{\partial y_i(n)}{\partial w_{kl}(n)} + \delta_{kj} \xi_l(n) \right].$$

б) Выведите алгоритм обучения для усиленной учителем рекуррентной сети.

Алгоритм несвязной расширенной фильтрации Калмана

15.15. Опишите, как можно использовать алгоритм DEKF для обучения простой рекуррентной сети, показанной на рис. 15.3. Для этого обучения можно также воспользоваться алгоритмом BPTT.

15.16. В своей обычной форме обучение DEKF последовательно корректирует синаптические веса. В противоположность стандартному обратному распространению выполняются простые коррекции градиентов, которые позволяют выбрать, применять ли их немедленно или накапливать в течение определенного времени и затем применять суммарную коррекцию. Несмотря на то что такое аккумулирование можно выполнять в самом алгоритме DEKF, это может привести к рассогласованности между вектором весов и матрицей ковариации ошибок, которая корректируется каждый раз, когда выполняется итерация по коррекции весов. Получается, что обучение DEKF препятствует пакетной коррекции весов. Однако в этом случае можно использовать *многопоточное обучение DEKF* (multistream DEKF training), которое позволяет существовать нескольким последовательностям обучения и при этом остается согласованным по отношению к теории фильтров Калмана [291], [292].

а) Рассмотрите эту задачу обучения с $N_{\text{вх.}}$ входами, $N_{\text{вых.}}$ выходами и фиксированным набором N примеров обучения. Из примеров обучения сформируйте $M \leq N$ потоков данных, которые питают M сетей, имеющих идентичные веса. В каждом цикле обучения один образец из каждого потока представляется соответствующей сети, и вычисляется $N_{\text{вых.}}$ выходных сигналов для каждого потока. После этого проводится единовременная коррекция, которая применяется к весам всех сетей. Выведите многопоточную форму алгоритма DEKF.

б) В качестве примера рассмотрите стандартную задачу XOR с четырьмя примерами обучения. Предполагается, что мы располагаем сетью прямого распространения, которая дополнена памятью с задержкой в линии, присоединенной к выходному слою. Таким образом, в результате получим четыре выхода сети: фактический выход сети, питающий память с задержкой в линии, и три задержанных его версии, каждую из которых также считаем новым выходом сети. Теперь в некотором порядке применим к этой сетевой структуре все четыре примера обучения, при этом не выполняя коррекцию весов. После представления четвертого примера мы получим четыре выхода сети, которые представляют обработку четырех примеров обучения сетью с одинаковыми весами. Если на этой основе мы выполним единовременную коррекцию вектора весов по алгоритму DEKF на основе данных четырех входов и четырех выходов сети, то получим задачу четырех потоков. Проверьте правильность этого примера.

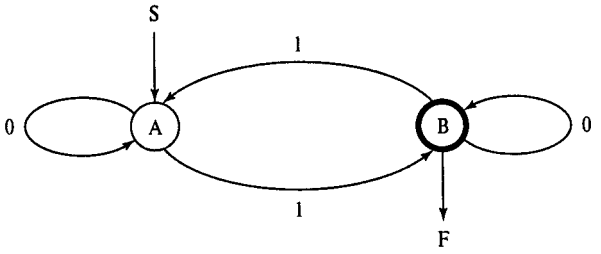


Рис. 15.23. Автомат с двумя состояниями

Рекуррентные сети второго порядка

15.17. В данном примере исследуем конструкцию *конечного автомата четности* (parity finite-state automata), использующего рекуррентную сеть второго порядка. Этот автомат распознает четное количество единиц в последовательной строке нулей и единиц конечной произвольной длины.

На рис. 15.23 показан автомат, принимающий два состояния. Эти состояния обозначены окружностями, а переходы — стрелками. Символом S обозначено состояние старта — в данном случае A. Тонкая окружность означает, что в этом состоянии (на рисунке это состояние B) мы принимаем строку. Автомат начинает сканировать строку в состоянии A и возвращается назад в состояние A, если обнаружен нуль, или переходит в состояние B, если обнаружена единица. Аналогично, находясь в состоянии B, он остается в этом же состоянии, если обнаружит нуль, и переходит в состояние A, если обнаружит единицу. Таким образом, данный автомат всегда будет находиться в состоянии A, если просканировано четное количество единиц (в том числе нуль штук), или в состоянии B, если количество единиц не четно.

Формализуя, определим множество состояний как $Q = \{A, B\}$, $S = A$ (начальное состояние), входной алфавит — как $\Sigma = \{0, 1\}$, состояние приема — как $F = B$ и функции перехода состояний — как:

$$\delta(A, 0) = A,$$
$$\delta(A, 1) = B,$$
$$\delta(B, 0) = B,$$
$$\delta(B, 1) = A.$$

Эти уравнения необходимы для применения формулы (15.9), относящейся к рекуррентной сети второго порядка. Более детально конечные автоматы описаны в [474].

Закодируйте приведенные выше правила перехода в рекуррентной сети второго порядка.

15.18. В разделе 15.8 выведен алгоритм рекуррентного обучения в реальном времени (RTRL) для полносвязной рекуррентной сети, использующей нейроны первого порядка. В разделе 15.2 описаны рекуррентные сети, использующие нейроны второго порядка.

Опровергните теорию, описанную в разделе 15.8, выведя алгоритм обучения для рекуррентной сети второго порядка.

Заключение

Предметная область нейронных сетей лежит на пересечении многих наук. Ее корни уходят в нейробиологию, математику, статистику, физику, науку о компьютерах и инженерию. Это видно из разнообразия вопросов, рассмотренных в настоящей книге. Способность нейронных сетей *обучаться* на данных с помощью учителя или без такового обеспечивает их важное свойство. Это свойство обучаемости имеет важное теоретическое и практическое применение. В той или иной форме способность нейронных сетей обучаться на примерах (представительных для своей среды) сделала их неоценимым инструментом в таких разнообразных областях применения, как моделирование, анализ временны*х* рядов, распознавание образов, обработка сигналов и управление. В частности, нейронным сетям есть что предложить, когда решение интересующей задачи становится сложным по одной из следующих причин.

- Из-за отсутствия физического либо статистического понимания системы.
- Из-за статистического разброса наблюдаемых данных.
- Из-за нелинейности механизма, ответственного за обобщение данных.

Новая волна интереса к нейронным сетям (начавшаяся во второй половине 1980-х годов) возникла вследствие того, что потребовалось обучение на нескольких уровнях. Алгоритмы обучения нейронных сетей позволили избежать необходимости в ручном извлечении признаков при распознавании рукописного текста. Градиентные алгоритмы обучения нейронных сетей позволили обучать системы извлечения признаков, классификаторы и контекстные процессоры (скрытые модели Маркова и языковые модели).

Обучение наполняет все уровни интеллектуальных систем во все увеличивающемся количестве областей применения. Исходя из этого, целесообразно завершить данную книгу некоторыми замечаниями относительно интеллектуальных систем и роли нейронных сетей в их создании.

Интеллектуальные системы

Поскольку мы не согласны с научным определением интеллекта[1] и в связи с ограниченным объемом данной книги, мы не будем углубляться в дискуссию относительно того, что такое интеллект. Вместо этого мы ограничимся кратким обзором интеллектуальных систем в контексте трех специфичных областей их применения: классификация образов, управление и обработка сигналов. Известно, что не существует "универсальных" интеллектуальных систем. Эти системы особенны для каждой конкретной предметной области их применения.

Большая часть усилий исследователей нейронных сетей была сфокусирована на задаче распознавания образов. Учитывая практическую важность этой задачи и ее повсеместную природу, а также тот факт, что нейронные сети исключительно хорошо подходят для решения задачи классификации, такая концентрация усилий ученых направлялась на поиск средств корректной классификации. Развивая это направление, стало возможным заложить основы *адаптивной классификации образов* (adaptive pattern classification). Однако мы достигли той точки, в которой системы классификации должны рассматриваться в более широком смысле, если мы хотим решать задачи классификации более сложной и интеллектуальной природы. На рис. 16.1 показана структура "гипотетической" системы классификации [413]. Первый уровень этой системы получает сенсорные данные, генерируемые некоторым источником информации. Второй уровень извлекает множество признаков, характеризующее полученные сенсорные данные. Третий уровень классифицирует эти признаки в одну или несколько различных категорий, которые затем помещаются в глобальный контекст на четвертом уровне. В заключение можно для примера поместить разобранный входной сигнал в некоторую форму базы данных, которую будет использовать конечный пользователь.

Важными признаками, характеризующими эту систему, являются следующие.

- *Распознавание* (recognition), являющееся результатом прямого прохождения информации от одного уровня системы к следующему в традиционной системе классификации.
- *Фокусировка* (focusing), при которой более высокий уровень системы способен избирательно влиять на обработку информации более низким уровнем с помощью знаний, накопленных на основе имеющихся данных.

Таким образом, новаторство системы классификации образов, показанной на рис. 16.1, лежит в *знаниях о целевой области* и их использовании нижними уровнями системы для повышения общей производительности системы при имеющихся фундаментальных ограничениях объема обрабатываемой информации. Будем надеяться,

[1] Философская дискуссия относительно понятия интеллекта, рассматриваемого с различных точек зрения, содержится в [8], [15], [592].

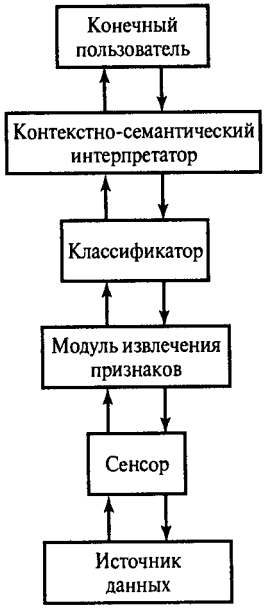


Рис. 16.1. Функциональная архитектура интеллектуальной системы

что классификация образов, использующая нейронные сети, будет развиваться в направлении создания моделей, которые будут последовательно наполняться знаниями о целевой области. Можно прогнозировать появление нового класса интеллектуальных систем распознавания образов, которые обеспечат следующие возможности.

- Способность извлекать *контекстные знания* (contextual knowledge) и использовать их с помощью *фокусировки* (focusing).
- *Локализованность*, а не распределенность представления знаний.
- *Разреженная архитектура* (sparse architecture), подчеркивающая модульность и иерархию как принцип конструкции сетей.

Реализация такой интеллектуальной системы может быть достигнута только путем комбинирования нейронных сетей с другими соответствующими средствами. Полезным инструментом, который приходит на ум, является *алгоритм Витерби* (Viterbi algorithm) — одна из форм динамического программирования, созданная для *последовательной* обработки информации[2], являющейся врожденной характеристикой системы, показанной на рис. 16.1. (Подробно об алгоритме динамического программирования см. в главе 12).

---

[2] Алгоритм Витерби впервые был изложен в [1096] и предназначен для решения задачи декодирования свертки в теории связи. Дополнительная информация об алгоритме Витерби содержится в [303].

Применение в задачах классификации образов, которое включает в себя комбинированное использование сетей свертки (см. главу 4) и алгоритма Витерби, описывается в [622], [623].

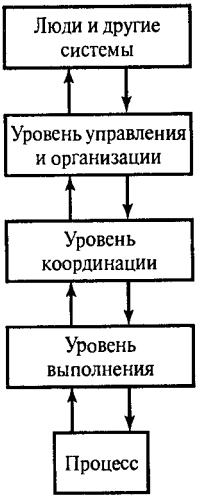


Рис. 16.2. Функциональная архитектура интеллектуальной системы управления

Управление — еще одна естественная область применения нейронных сетей — также развивалось в направлении *интеллектуального управления*[3]. Автономность является важной целью разработчиков систем управления, и интеллектуальные контроллеры рассматриваются как один из путей ее достижения. На рис. 16.2 показана функциональная архитектура интеллектуального автономного контроллера [65], [820]. Эта система имеет следующие функциональные уровни.

1. *Уровень выполнения* (execution level). Включает низкоуровневую обработку сигнала и алгоритмы адаптивного управления и идентификации.
2. *Уровень координации* (coordination level). Реализует связь уровней выполнения и управления, контролируя такие вопросы, как *настройка* (tuning), *контроль* (supervision), *разрешение кризисов* (crisis management) и *планирование* (planning).
3. *Уровень управления и организации* (management and organization level). Осуществляет контроль над низкоуровневыми функциями и управление интерфейсом с человеком.

В то время как классическое управление уходит корнями в теорию линейных дифференциальных уравнений, интеллектуальное управление *основывается на правилах* (rule based), так как зависимости, вовлеченные в его реализацию, настолько сложны, что не допускают аналитического представления. Для того чтобы работать с такими зависимостями, целесообразно использовать математику нечетких систем и нейрон-

[3] Интеллектуальное управление рассматривается в [403], [1065], [1130].

ных сетей. Сила *нечетких систем*[4] (fuzzy system) заключена в их способности создавать количественное представление лингвистического входа и быстро предоставлять эффективную аппроксимацию сложных и зачастую неизвестных правил отображения входа на выход в различных системах. Сила нейронных сетей заключается в их способности *обучаться* на данных. Между нейронными сетями и нечеткими системами существует определенная синергетика, которая делает их гибридизацию важным инструментом интеллектуального управления и других приложений.

Переходя далее к обработке сигналов, можно сказать, что не существует более плодородной области применения нейронных сетей и, в частности, их характеристик нелинейности и адаптивности [434]. Многие физические явления отвечают за генерацию *информационных сигналов* (information-bearing signal), встречаемых на практике (в частности, речевых, радарных и сонарных сигналов), управляются *нелинейной динамикой нестационарной и сложной природы*, которая не имеет точного математического описания. Для использования полного информационного содержания таких сигналов во все моменты времени требуются интеллектуальные машины обработки сигнала[5], конструкция которых решает следующие ключевые вопросы.

- *Нелинейность* (nonlinearity), которая делает возможным извлечение статистики высокого порядка из входного сигнала.
- *Обучение и адаптация* (learning and adaptation), посредством которой система может обучиться исследуемому физическому механизму среды и адаптироваться к непрерывным медленным статистическим вариациям внешней среды.
- *Механизм внимания* (attentional mechanism), в котором посредством взаимодействия с конечным пользователем или с помощью самоорганизации система может сфокусировать свои вычислительные мощности на конкретной области изображения или на конкретном месте в пространстве для их более детального анализа[6].

---

[4] Теория нечетких множеств была заложена в [1175], [1176] с целью реализации математического инструментария работы с лингвистическими переменными (т.е. концепциями, описанными на естественном языке). Толкование нечеткой логики в книжной форме содержится в [268]. В [591] принимается другая точка зрения: нечеткие системы рассматриваются как аппроксиматоры функций. В ней показано, что нечеткая система способна моделировать любую непрерывную функцию или систему при условии использования достаточных правил.

[5] Специальный выпуск статей IEEE 1998 (Institute of Electrical and Electronic Engineers) был посвящен вопросу интеллектуальной обработки сигналов [440].

[6] Самоорганизующиеся системы для иерархической фокусировки или избирательного внимания были описаны в [324]. Описанная система являлась модификацией многослойного неокогнитрона

Самоорганизующийся механизм внимания также представлен в *теории адаптивного резонанса* (adaptive resonance theory), основы которой были заложены в [175], [177]. Эта теория, применяемая к задаче адаптивного распознавания образов, использует комбинацию фильтрации снизу вверх (bottom-up filtering) и сравнения образцов сверху вниз (top-down template matching).

На рис. 16.3 показана функциональная архитектура интеллектуальных машин обработки сигнала, которая имеет три уровня.

1. *Низкоуровневая обработка* (low-level processing), целью которой является предварительная обработка полученного сигнала и его подготовка для второго уровня. В предварительную подготовку входит использование фильтрации для уменьшения эффекта шума, а также другие продвинутые операции обработки сигнала, такие как *частотно-временной анализ*[7] (time-frequency analysis). Целью частотно-временного анализа является описание того, как развивается спектральное содержание сигнала и каков спектр, изменяющийся во времени. В частности, одномерное (временное) представление получаемого сигнала преобразовывается в двумерное изображение, одно измерение которого является частотой, а второе — временем. Частотный анализ реализует эффективный метод выявления нестационарной природы получаемого сигнала, который делает ее более различимой, чем исходная временная форма.
2. *Уровень обучения и адаптации* (learning and adaptation level), на котором память (как краткосрочная, так и долгосрочная) и механизм внимания встраиваются в конструкцию системы. Например, в многослойном персептроне, подвергающемся обучению с учителем на большом множестве данных, представительном для среды функционирования системы, общая статистическая информация о среде хранится в синаптических весах сети. Для того чтобы учесть медленные статистические вариации среды во времени, к выходу многослойного персептрона добавляется схема слепой адаптации (т.е. подсистема непрерывного обучения, работающая без учителя). Этот процесс обучения также включает обеспечение *сети внимания*[8] (attentional network), в которой система может сфокусировать свое внимание на особенно важных свойствах получаемого сигнала, "вентилируя" поток информации от нижнего слоя к верхним, если возникнет такая потребность.
3. *Уровень принятия решения* (decision-making level), на котором система принимает окончательное решение. Решение может приниматься относительно того, содержится ли во входном сигнале интересующий объект, как в случае радара или сонара, и как интерпретировать поступившую информацию — как единицу или нуль (в случае цифровой связи). В процессе принятия решения также учитываются уровни доверия.

---

[7] Детальное описание частотно-временного анализа, построенного на классической теории Фурье, содержится в [199].

Теория и приложения распределения Вигнера (Wigner distribution), важного инструмента в билинейном/квадратичном частотно-временном представлении, описаны в [723].

Еще одна точка зрения, в которой вместо частоты мы рассматриваем амплитуду (scale), представлена в работе, посвященной элементарным волнам и связанным с ними вопросам кодирования полосы (subband coding) [1093].

[8] В [1078] описана нейросетевая модель избирательного сокрытия визуального внимания (selective covert visual attention). Эта модель может обучиться фокусированию внимания на характеристиках, важных для конкретной решаемой задачи, с помощью модуляции потока информации на предварительной стадии.

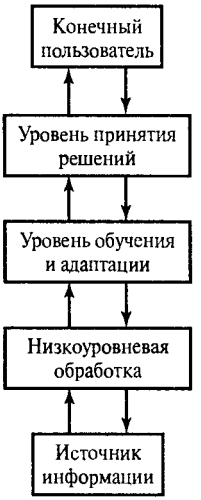


Рис. 16.3. Функциональная архитектура интеллектуальных машин обработки сигнала

Мы не заявляем, что описанные здесь системы являются единственным способом встраивания интеллекта в системы классификации образов, управления и обработки сигнала. Они представляют систематические способы достижения этой цели. Несмотря на различия в терминологии различных областей применения, они обладают некоторыми общими признаками.

- Существует двунаправленный поток информации, от нижнего слоя к верхнему, и наоборот.
- Более высокие уровни часто связаны с теми аспектами динамики системы, которые требуют длительной обработки, которые шире по объему информации (scope) или имеют более дальний горизонт.
- При перемещении от нижнего слоя к верхнему наблюдается повышение интеллекта наряду с понижением точности.
- На более высоких уровнях понижается *зернистость* (granularity) (т.е. повышается уровень абстракции).

В главе 1 рассмотрение вопросов нейронных сетей было начато с описания человеческого мозга — источника мотивации нейронных сетей — как гигантской системы обработки информации. Поэтому мы и завершаем книгу кратким описанием интеллектуальных машин, которые осуществляют интеллектуальную обработку информации. Борьба за создание интеллектуальных машин продолжается.

Библиография

1. Aarts E. and J. Korst. Simulated Annealing and Boltzmann Machines: A Stochastic Aproach to Combinatorial Optimization and Neural Computing, New York: Wiley, 1989.
2. Abarbanel H.D.I. Analysis of Observed Chaotic Data, New York: Springer-Verlag, 1996.
3. Abraham R.H. and C.D. Shaw. Dynamics of the Geometry of Behavior, Reading, MA: Addison-Wesley, 1992.
4. Abu-Mostafa Y.S. "Hints", Neural computation, 1995, vol. 7, p. 639–671.
5. Abu-Mostafa Y.S. "Learning from hints in Neural Networks", Journal of Complexity, 1990, vol. 6, p. 192–198.
6. Abu-Mostafa Y.S. "The Vapnik-Chervonenkis Dimension: Information Versus Complexity in Learning", Neural Computation, 1989, vol. 1, p. 312–317.
7. Abu-Mostafa Y.S. and J.M. St. Jacques. "Information capacity of the Hopfield model", IEEE Transactions on Information Theory, 1985, vol. IT-31, p. 461–464.
8. Ackerman P.L. "Intelligence", In S.C. Shapiro, ed. Encyclopedia of Artificial Intelligence, New York: Wiley (Interscience) 1990, p. 431–440.
9. Ackley D.H., G.E. Hinton and T.J. Sejnowski. "A Learning Algorithm for Boltzmann Machines", Cognitive Science, 1985, vol. 9, p. 147–169.
10. Aiyer S.V.B., N. Niranjan and F. Fallside. "A theoretical investigation into the performance of the Hopfield model", IEEE Transactions on Neural Networks, 1990, vol. 15, p. 204–215.
11. Aizerman M.A., E.M. Braverman and L.I. Rozonoer. "Theoretical foundations of the potential function method in pattern recognition learning", Automation and Remote Control, 1964, vol. 25, p. 821–837.
12. Aizerman M.A., E.M. Braverman and L.I. Rozonoer. "The probability problem of pattern recognition learning and the method of potential functions", Automation and Remote Control, 1964Б, vol. 25, p. 1175–1193.
13. Akaike H. "A new look at the statistical model identification", IEEE Transactions on Automatic Control, 1974, vol. AC-19, p. 716–723.
14. Akaike H. "Statistical predictor identification", Annals of the Institute of Statistical Mathematics, 1970, vol. 22, p. 202–217.
15. Albus J.S. "Outline for a theory of intelligence", IEEE Transactions on Systems, Man and Cybernetics, 1991, vol. 21, p. 473–509.
16. Aleksander I. and H. Morton. An Introduction to Neural Computing, London: Chapman and Hall, 1990.
17. Allport A. "Visual attention", In Foundations of Cognitive Science, M.I. Posner, ed., 1989, p. 631–682, Cambridge, MA: MIT Press.

18. Al-Mashoug K.A. and I.S. Reed. "Including hints in training neural nets", Neural Computation, 1991, vol. 3, p. 418–427.
19. Alspector J., R.B. Allen, A. Jayakumar, T. Zepenfeld and R. Meir. "Relaxation networks for large supervised learning problems", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 1015–1021, San Mateo, CA: Morgan Kaufmann.
20. Alspector J., A. Jayakumar and S. Luna. "Experimental evaluation of learning in a neural microsystem", Advances in Neural Information Processing Systems, 1992, vol. 4, p. 871–878, San Mateo, CA: Morgan Kaufmann.
21. Alspector J., R. Meir, B. Yuhas, A. Jayakumar and D. Lipe. "A parallel gradient descent method for learning in analog VLSI neural networks", Advances in Neural Information Processing Systems, 1993, vol. 5, p. 836–844, San Mateo, CA: Morgan Kaufmann.
22. Amari S. "Natural gradient works efficiently in learning", Neural Computation, 1998, vol. 10, p. 251–276.
23. Amari S. Private communication, 1997.
24. Amari S. "A universal theorem on learning curves", Neural Networks, 1993, vol. 6, p. 161–166.
25. Amari S. "Mathematical foundations of neurocomputing", Proceedings of the IEEE, 1990, vol. 78, p. 1443–1463.
26. Amari S. "Differential geometry of a parametric family of invertible systems — Riemanian metric, dual affine connections and divergence", Mathematical Systems Theory, 1987, vol. 20, p. 53–82.
27. Amari S. Differential-Geometrical Methods in Statistics, New York: Springer-Verlag, 1985.
28. Amari S. "Field theory of self-organizing neural nets", IEEE Transactions on Systems, Man and Cybernetics, 1983, vol. SMC-13, p. 741–748.
29. Amari S. "Topographic organization of nerve fields", Bulletin of Mathematical Biology, 1980, vol. 42, p. 339–364.
30. Amari S. "Neural theory of association and concept-formation", Biological Cybernetics, 1977, vol. 26, p. 175–185.
31. Amari S. "Dynamics of pattern formation in lateral-inhibition type neural fields", Biological Cybernetics, 1977Б, vol. 27, p. 77–87.
32. Amari S. "Characteristics of random nets of analog neuron-like elements", IEEE Transactions on Systems, Man and Cybernetics, 1972, vol. SMC-2, p. 643–657.
33. Amari S. "A theory of adaptive pattern classifiers", IEEE Trans. Electronic Computers, 1967, vol. EC-16, p. 299–307.
34. Amari S. and M.A. Arbib. "Competition and cooperation in neural nets", in J. Metzler, ed., Systems Neuroscience, 1977, p. 119–165, New York: Academic Press.

35. Amari S. and J.-F. Cardoso. "Blind source separation — Semiparametric statistical aproach", IEEE Transactions on Signal Processing, 1997, vol. 45, p. 2692–2700.

36. Amari S., T.-P. Chen and A. Cichoki. "Stability analysis of learning algorithms for blind source separation", Neural Networks, 1997, vol. 10, p. 1345–1351.

37. Amari S., A. Cichoki and H.H.Yang. "A new learning algorithm for blind signal separation", Advances in Neural Information Processing Systems, 1996, vol. 8, p. 757–763, Cambridge, MA: MIT Press.

38. Amari S. and K. Maginu. "Statistical neurodynamics of associative memory", Neural Networks, 1988, vol. 1, p. 63–73.

39. Amari S., K. Yoshida and K.-I. Kanatani. "A mathematical foundation for statistical neurodynamics", SIAM Journal of Aplied Mathematics, 1977, vol. 33, p. 95–126.

40. Amari S., N. Murata, K.-R. M(ller, M. Finke and H. Yang. "Statistical theory of overtraining — Is cross-validation asymptotically effective"? Advances in Neural Information Processing Systems, 1996a, vol. 8, p. 176–182, Cambridge, MA: MIT Press.

41. Ambros-Ingerson J., R. Granger and G. Lynch. "Simulation of paleo-cortex performs hierarchical clustering", Science, 1990, vol. 247, p. 1344–1348.

42. Amit D.J. Modeling Brain Function: The World of Attractor Neural Networks, New York: Cambridge University Press, 1989.

43. Anastasio T.J. "Vestibulo-ocular reflex: Performance and plasticity", In M.A. Arbib, ed., The Handbook of Brain Theory and Neural Networks, Cambridge, MA: MIT Press, 1995.

44. Anastasio T.J. "Modeling vestibulo-ocular reflex dynamics: From classical analysis to neural networks", in F. Eeckman, ed., Neural Systems: Analysis and Modeling, 1993, p. 407–430, Norwell, MA: Kluwer.

45. Anastasio T.J. "A recurrent neural network model of velocity storage in the vestibulo-ocular reflex", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 32–38, San Mateo, CA: Morgan Kaufmann.

46. Anderson J.A. Introduction to Neural Networks, Cambridge, MA: MIT Press, 1995.

47. Anderson J.A. "The BSB model: A simple nonlinear autoassociative neural network", in Associative Neural Memories (M. Hassoun, ed.), 1993, p. 77–103, Oxford: Oxford University Press.

48. Anderson J.A. "General introduction", Neurocomputing: Foundations of Research (J.A. Anderson and E. Rosenfeld, eds.), 1988, p. xiii-xxi, Cambridge, MA: MIT Press.

49. Anderson J.A. "Cognitive and psychological computation with neural models", IEEE Transactions on Systems, Man and Cybernetics, 1983, vol. SMC-13, p. 799–815.

50. Anderson J.A. "A simple neural network generating an interactive memory", Mathematical Biosciences, 1972, vol. 14, p. 197–220.

51. Anderson J.A. and G.L. Murphy. "Concepts in connectionist models", in Neural Networks for Computing, J.S. Denker, ed., 1986, p. 17–22, New York: American Institute of Physics.

52. Anderson J.A. and E. Rosenfeld, eds. Neurocomputing: Foundations of Research, Cambridge, MA: MIT Press, 1988.

53. Anderson J.A., A. Pellionisz and E. Rosenfeld, eds. Neurocomputing 2: Directions for Research, Cambridge, MA: MIT Press, 1990a.

54. Anderson J.A., J.W. Silverstein, S.A. Ritz and R.S. Jones. "Distinctive features, categorical perception and probability learning: Some aplications of a neural model", Psychological Review, 1977, vol. 84, p. 413–451.

55. Anderson J.A. and J.P. Sutton. "A network of networks: Computation and neurobiology", World Congress on Neural Networks, 1995, vol. I, p. 561–568.

56. Anderson J.A., M.T. Gately, P.A. Penz and D.R. Collins. "Radar signal categorization using a neural network", Proceedings of the IEEE, 1990Б, vol. 78, p. 1646–1657.

57. Anderson T.W. An Introduction to Multivariate Statistical Analysis, 2nd edition, New York: Wiley, 1984.

58. Andreou A.G. "On physical models of neural computation and their analog VLSI implementation", Proceedings of the 1994 Workshop on Physics and Computation, IEEE Computer Society Press, 1994, p. 255–264, Los Alamitos, CA.

59. Andreou A.G., K.A. Boahen, P.O. Pouliqueen, A. Pasavoic, R.E. Jenkins and K. Strohbehn. "Current-mode subthreshold MOS circuits for analog VLSI neural systems", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 205–213.

60. Andreou A.G., R.C. Meitzler, K. Strohbehn and K.A. Boahen. "Analog VLSI neuromorphic image acquisition and pre-processing systems", Neural Networks, 1995, vol. 8, p. 1323–1347.

61. Andrews R. and J. Diederich, eds. Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop, University of Sussex, Brighton, UK., 1996.

62. Ansaklis P.J., M. Lemmon and J.A. Stiver. "Learning to be autonomous", In M.D. Gupta and N.K. Sinha, eds., Intelligent Control Systems, 1996, p. 28–62, New York: IEEE Press.

63. Ansari N. and E. Hou. Computational Intelligence for Optimization, Norwell, MA: Kluwer, 1997.

64. Anthony M. and N. Biggs. Computational Learning Theory, Cambridge: Cambridge University Press, 1992.
65. Antsaklis P.J. and K.M. Passino, eds. An Introduction to Intelligent and Automatic Control, Norwell, MA: Kluwer, 1993.
66. Arbib M.A. The Metaphorical Brain, 2nd edition, New York: Wiley, 1989.
67. Arbib M.A. Brains, Machines and Mathematics, 2nd edition, New York: Springer-Verlag, 1987.
68. Arbib M.A., ed. The Handbook of Brain Theory and Neural Networks, Cambridge, MA: MIT Press, 1995.
69. Arrowsmith,D.K. and C.M. Place. An Introduction to Dynamical Systems, Cambridge: Cambridge University Press, 1990.
70. Artola A. and W. Singer. "Long-term potentiation and NMDA receptors in rat visual cortex", Nature, 1987, vol. 330, p. 649–652.
71. Ash R.E. Information Theory, New York: Wiley, 1965.
72. Ashby W.R. Design for a Brain, 2nd edition, New York: Wiley.
73. Ashby W.R., 1952. Design for a Brain, New York: Wiley, 1960.
74. Aspray W. and A. Burks. Papers of John von Neumann on Computing and Computer Theory, Charles Babbage Institute Reprint Series for the History of Computing, 1986, vol. 12. Cambridge, MA: MIT Press.
75. Åström K.J. and T.J. McAvoy. "Intelligent control: An overview and evaluation", Handbook of Intelligent Control, D.A. White and D.A. Sofge, eds., New York: Van Nostrand Reinhold, 1992.
76. Atherton D.P. Stability of Nonlinear Systems, Chichester, UK: Research Studies Press, 1981.
77. Atick J.J. "Could information theory provide an ecological theory of sensory processing"? Network: Computation in Neural Systems, 1992, vol. 3, p. 213–251.
78. Atick J.J. and A.N. Redlich. "What does the retina know about natural scenes", Neural Computation, 1992, vol. 4, p. 196–210.
79. Atick J.J. and A.N. Redlich. "Towards a theory of early visual processing", Neural Computation, 1990, vol. 2, p. 308–320.
80. Atick J.J., P.A. Griffin and A.N. Redlich. "Statistical aproach to shape from shading: Reconstruction of three-dimensional face surfaces from single two-dimensional images", Neural Computation, 1996, vol. 8, p. 1321–1340.
81. Atiya A.F. "Learning on a general network", Neural Information Processing Systems, D.Z. Anderson, ed., 1987, p. 22–30, New York: American Institute of Physics.
82. Atiya A.F. and Y.S. Abu-Mostafa. "An analog feedback associative memory", IEEE Transactions on Neural Networks, 1993, vol. 4, p. 117–126.

83. Attneave F. "Some informational aspects of visual perception", Psychological Review, 1954, vol. 61, p. 183–193.
84. Back A.D. and A.S. Weigend. "A first aplication of independent component analysis to extracting structure from stock returns", International Journal of Neural Systems, 1998, vol. 9, Special Issue on Data Mining in Finance.
85. Back A.D. and A.C. Tsoi. "FIR and IIR synapses, a new neural network architecture for time series modeling", Neural Computation, 1991, vol. 3, p. 375–385.
86. Back A.D. and A.C. Tsoi. "A low-sensitivity recurrent neural network", Neural Computation, 1998, vol. 10, p. 165–188.
87. Baldi P. and K. Hornik. "Neural networks and principal component analysis: Learning from examples without local minimum", Neural Networks, 1989, vol. 1, p. 53–58.
88. Bantine W.L. and A.S. Weigend. "Computing second derivatives in feed-forward networks: A review", IEEE Transactions on Neural Networks, 1994, vol. 5, p. 480–488.
89. Baras J.S. and A. LaVigna. "Convergence of Kohonen's learning vector quantization", International Joint Conference on Neural Networks, 1990, vol. III, p. 17–20, San Diego, CA.
90. Barlow H.B. "Unsupervised learning", Neural Computation, 1989, vol. 1, p. 295–311.
91. Barlow H.B. "Cognitronics: methods for acquiring and holding cognitive knowledge", 1985.
92. Barlow H.B. "Sensory mechanisms, the reduction of redundancy and intelligence", The Mechanisation of Thought Processes, National Physical Laboratory Symposium No. 10, Her Majesty's Stationary Office, London, 1959.
93. Barlow H. and P. F(ldi(k. "Adaptation and decorrelation in the cortex", in The Computing Neuron, R. Durbin, C. Miall and G. Mitchison, eds., 1989, p. 54–72. Reading, MA: Addison-Wesley.
94. Barnard E. and D. Casasent. "Invariance and neural nets", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 498–508.
95. Barron A.R. "Neural net aproximation", in Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems, 1992, p. 69–72, New Haven, CT.: Yale University.
96. Barron A.R. "Universal aproximation bounds for superpositions of a sigmoidal function", IEEE Transactions on Information Theory, 1993, vol. 39, p. 930–945.
97. Bartlett P.L. "For valid generalization, the size of the weights is more important than the size of the network", Advances in Neural Information Processing Systems, 1997, vol. 9, p. 134–140, Cambridge, MA: MIT Press.

98. Barto A.G. "Reinforcement learning and adaptive critic methods", Handbook of Intelligent Control, D.A. White and D.A. Sofge, eds., 1992, p. 469–491, New York: Van Nostrand Reinhold.

99. Barto A.G., S.J. Bradtke and S. Singh. "Learning to act using real-time dynamic programming", Artificial Intelligence, 1995, vol. 72, p. 81–138.

100. Barto A.G., R.S. Sutton and C.W. Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems", IEEE Transactions on Systems, Man and Cybernetics, 1983, vol. SMC-13, p. 834–846.

101. Basar E., ed. Chaos in Brain Function, New York: Springer-Verlag, 1990.

102. Bashkirov O.A., E.M. Braverman and I.B. Muchnik. "Potential function algorithms for pattern recognition learning machines", Automation and Remote Control, 1964, vol. 25, p. 629–631.

103. Battiti R. "First- and second-order methods for learning: Between steepest descent and Newton's method", Neural Computation, 1992, vol. 4, p. 141–166.

104. Bauer H.-U. and K.R. Pawelzik. "Quantifying the neighborhood preservation of self-organizing feature maps", IEEE Transactions on Neural Networks, 1992, vol. 3, p. 570–579.

105. Bauer H.-U., R. Der and M. Hermman. "Controlling the magnification factor of self-organizing feature maps", Neural Computation, 1996, vol. 8, p. 757–771.

106. Baum E.B. "Neural net algorithms that learn in polynomial time from examples and queries", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 5–19.

107. Baum E.B. and D. Haussler. "What size net gives valid generalization"? Neural Computation, 1989, vol. 1, p. 151–160.

108. Baum E.B. and F. Wilczek. "Supervised learning of probability distributions by neural networks", in D.Z. Anderson, ed., 1988, p. 52–61, New York: American Institute of Physics.

109. Beaufays F. and E.A. Wan. "Relating real-time backpropagation and backpropagation-through-time: An aplication of flow graph interreciprocity", Neural Computation, 1994, vol. 6, p. 296–306.

110. Becker S. "Mutual information maximization: models of cortical self-organization", Network: Computation in Neural Systems, 1996, vol. 7, p. 7–31.

111. Becker S. "Learning to categorize objects using temporal coherence", Advances in Neural Information Processing Systems, 1993, vol. 5, p. 361–368, San Mateo, CA: Morgan Kaufmann.

112. Becker S. "Unsupervised learning procedures for neural networks", International Journal of Neural Systems, 1991, vol. 2, p. 17–33.

113. Becker S. and G.E. Hinton. "Learning mixture models of spatial coherence", Neural Computation, 1993, vol. 5, p. 267–277.

114. Becker S. and G.E. Hinton. "A self-organizing neural network that discovers surfaces in random-dot stereograms", Nature (London), 1992, vol. 355, p. 161–163.

115. Beckerman M. Adaptive Cooperative Systems, New York: Wiley (Interscience), 1997.

116. Bell A.J. and T.J. Sejnowski. "An information-maximization aproach to blind separation and blind deconvolution", Neural Computation, 1995, vol. 6, p. 1129–1159.

117. Bellman R. Adaptive Control Processes: A Guided Tour, Princeton, NJ: Princeton University Press, 1961.

118. Bellman R. Dynamic Programming, Princeton, NJ: Princeton University Press, 1957.

119. Bellman R. and S.E. Dreyfus. Aplied Dynamic Programming, Princeton, NJ: Princeton University Press, 1962.

120. Bengio Y. Neural Networks for Speech and Sequence Recognition. London: International Thomson Computer Press, 1996.

121. Bengio Y., P. Simard and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994, vol. 5, p. 157–166.

122. Benveniste A., M. Métivier and P. Priouret. Adaptive Algorithms and Stochastic Aproximation, New York: Springer-Verlag, 1987.

123. Bertero M., T.A. Poggio and V. Torre. "Ill-posed problems in early vision", Proceedings of the IEEE, 1988, vol. 76, p. 869–889.

124. Bertsekas D.P. Dynamic Programming and Optimal Control, 1995, vol. I and vol. II, Belmont, MA: Athenas Scientific.

125. Bertsekas D.P. Nonlinear Programming, Belmont, MA: Athenas Scientific, 1995.

126. Bertsekas D.P. and J.N. Tsitsiklis. Neuro-Dynamic Programming, Belmont, MA: Athenas Scientific, 1996.

127. Beymer D. and T. Poggio. "Image representations for visual learning", Science, 1996, vol. 272, p. 1905–1909.

128. Bienenstock E.L., L.N. Cooper and P.W. Munro. "Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex", Journal of Neuroscience, 1982, vol. 2, p. 32–48.

129. Bishop C.M. "Exact calculation of the Hessian matrix for the multi-layer perceptron", Neural Computation, 1992, vol. 4, p. 494–501.

130. Bishop C.M. Neural Networks for Pattern Recognition. Oxford: Clarendon Press, 1995.

131. Black I.B. Information in the Brain: A Molecular Perspective, Cambridge, MA: MIT Press, 1991.

132. Blake A. "The least-disturbance principle and week constraints", Pattern Recognition Letters, 1983, vol. 1, p. 393–399.

133. Bliss T.V.P. and T. Lomo. "Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforatant path", J. Physiol, 1973, vol. 232, p. 331–356.

134. Blumer A., A. Ehrenfeucht, D. Haussler and M.K. Warmuth. "Learnability and the Vapnik-Chervonenkis Dimension", Journal of the Association for Computing Machinery, 1989, vol. 36, p. 929–965.

135. Blumer A., A. Ehrenfeucht, D. Haussler and M.K. Warmuth. "Occam's razor", Information Processing Letters, 1987, vol. 24, p. 377–380.

136. Boahen K.A. "A retinomorphic vision system", IEEE Micro, 1996, vol. 16, no. 5, p. 30–39.

137. Boahen K.A. and A.G. Andreou. "A contrast sensitive silicon retina with reciprocal synapses", Advances in Neural Information Processing Systems, 1992, vol. 4, p. 764–772. San Mateo, CA: Morgan Kaufmann.

138. Boahen K.A., P.O. Pouliqueen, A.G. Andreou and R.E. Jenkins. "A heterassociative memory using current-mode analog VLSI circuits", IEEE Transactions on Circuits and Systems, 1989, vol. CAS-36, p. 747–755.

139. Bodenhausen U. and A. Waibel. "The tempo 2 algorithm: Adjusting time-delays by supervised learning", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 155–161, San Mateo, CA: Morgan Kaufmann.

140. Boltzmann L. "Weitere studien über das Wärmegleichgewicht unter gasmolekülen", Sitzungsberichte der Mathematisch-Naturwissenschaftlichen Classe der Kaiserlichen Akademie der Wissenschaften, 1872, vol. 66, p. 275–370.

141. Boser B., I. Guyon and V.N. Vapnik. "A training algorithm for optimal margin classifiers", Fifth Annual Workshop on Computational Learning Theory, 1992, p. 144–152. San Mateo, CA: Morgan Kaufmann.

142. Boser B.E., E. S(ckinger, J. Bromley, Y. LeCun and L.D. Jackel. "Hardware requirements for neural network pattern classifiers", IEEE Micro, 1992, vol. 12, p. 32–40.

143. Bourlard H.A. and N. Morgan. Connectionist Speech Recognition: A Hybrid Aproach, Boston: Kluwer, 1994.

144. Bourlard H.A. and C.J. Wellekens. "Links between Markov models sand multilayer perceptrons", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, vol. PAMI-12, p. 1167–1178.

145. Box G.E.P. and G.M. Jenkins. Time Series Analysis: Forecasting and Control, San Francisco: Holden Day, 1976.

146. Braitenberg V. "Is the cerebella cortex a biological clock in the millisecond range"? in The Cerebellum. Progress in Brain Research, C.A. Fox and R.S. Snider, eds., 1967, vol. 25, p. 334–346, Amsterdam: Elsevier.

147. Braitenberg V. "Reading the structure of brains", Network: Computation in Neural Systems, 1990, vol. 1, p. 1–12.

148. Braitenberg V. "Two views of the cerebral cortex", in Brain Theory, G. Palm and A. Aertsen, eds., 1986, p. 81–96. New York: Springer-Verlag.

149. Braitenberg V. Vehicles: Experiments in Synthetic Psychology, Cambridge, MA: MIT Press, 1984.

150. Braitenberg V. On the Texture of Brains, New York: Springer-Verlag, 1977.

151. Bregman A.S. Auditory Scene Analysis: The Perceptual Organization of Sound, Cambridge, MA: MIT Press, 1990.

152. Breiman L. "Bagging predictors", Machine Learning, 1996, vol. 24, p. 123–140.

153. Breiman L. "Bias, variance and arcing classifiers", Technical Report 460, Statistics Department, University of California, Berkeley, Calif, 1996Б.

154. Breiman L., J. Friedman, R. Olshen and C. Stone. Classification and Regression Trees, New York: Chapman and Hall, 1984.

155. Bridle J.S. "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition", Neuro-computing: Algorithms, Architectures and Aplications, F. Fougelman-Soulie and J. H(rault, eds., 1990, New York: Springer-Veriag.

156. Bridle J.S. "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters", Advances in Neural Information Processing Systems, 1990Б, vol. 2, p. 211–217, San Mateo, CA: Morgan Kaufmann.

157. Brodal A. Neurological Anatomy in Relation to Clinical Medicine, 3rd edition, New York: Oxford University Press, 1981.

158. Brodmann K. Vergleichende Lokalisationslehre der Grosshirnrinde, Leipzig: J.A. Barth, 1909.

159. Brogan W.L. Modern Control Theory, 2nd edition, Englewood Cliffs, NJ: Prentice-Hall, 1985.

160. Broomhead D.S. and D. Lowe. "Multivariable functional interpolation and adaptive networks", Complex Systems, 1988, vol. 2, p. 321–355.

161. Brown T.H., E.W. Kairiss and C.L. Keenan. "Hebbian synapses: Biophysical mechanisms and algorithms", Annual Review of Neuroscicnce, 1990, vol. 13, p. 475–511.

162. Bruck J. "On the convergence properties of the Hopfield model", Proceedings of the IEEE, 1990, vol. 78, p. 1579–1585.

163. Bryson A.E., Jr. and Y.C. Ho. Aplied Optimal Control, Blaisdell, 1969. (Вторая редакция, 1975, Hemisphere Publishing, Washington, DC).

164. Burg J.P. Modern Spectral Estimation, Ph.D. Thesis, Stanford University, Stanford, Calif, 1975.

165. Burges C.J.C. "A tutorial on suport vector machines for pattern recognition", Data Mining and Knowledge Discovery, 1998.

166. Cacoullos T. "Estimation of a multivariate density", Annals of the Institute of Statistical Mathematics (Tokyo), 1966, vol. 18, p. 179–189.

167. Caianiello E.R. "Outline of a theory of thought-processes and thinking machines", Journal of Theoretical Biology, 1961, vol. 1, p. 204–235.

168. Cameron S.H. Tech. Report 60-600, Proceedings of the Bionics Symposium, 1960, p. 197–212, Wright Air Development Division, Dayton, Ohio.

169. Cardoso J.-F. "Blind signal separation: A review", Proceedings of the IEEE, 1998a, vol. 86.

170. Cardoso J.-F. "Multidimensional independent component analysis", Proceedings IEEE ICASSP, Seattle, WA, May, 1998Б.

171. Cardoso J.-F. "Infomax and maximum likelihood for blind source separation", IEEE Signal Processing Letters, 1997, vol. 4, p. 112–114.

172. Cardoso J.-F. "Entropic contrasts for source separation", Presented at the NIPS 96 Workshop on Blind Signal Processing organized by A. Cichoki at Snomass, Colo. Издана как глава книги Unsupervised Adaptive Filtering, S. Haykin, ed., 1996, New York: Wiley.

173. Cardoso J.-F. and B. Laheld. "Equivariant adaptive source separation", IEEE Transactions on Signal Processing, 1996, vol. 44, p. 3017–3030.

174. Cardoso J.-F. and A. Souloumia. "Blind beamforming for non-Gaussian signals", IEE Proceedings (London), Part F, 1993, vol. 140, p. 362–370.

175. Carpenter G.A. and S. Grossberg. "A massively parallel architecture for a self-organizing neural pattern recognition machine", Computer Vision, Graphics and Image Processing, 1987, vol. 37, p. 54–115.

176. Carpenter G.A., M.A. Cohen and S. Grossberg. Technical comments on "Computing with neural networks", Science, 1987, vol. 235, p. 1226–1227.

177. Carpenter G.A. and S. Grossberg. "Adaptive resonance theory (ART)", in M.A. Arbib, 1995, ed., The Handbook of Brain Theory and Neural Networks, p. 79–82, Cambridge, MA: MIT Press.

178. Casdagli M. "Nonlinear prediction of chaotic time-series", Physica, 1989, vol. 35D, p. 335–356.

179. Černy V. "Thermodynamic aproach to the travelling salesman problem", Journal of Optimization Theory and Aplications, 1985, vol. 45, p. 41–51.

180. Changeux J.P. and A. Danchin. "Selective stabilization of developing synapses as a mechanism for the specification of neural networks", Nature, 1976, vol. 264, p. 705–712.

181. Chatterjee C., V.P. Roychowdhhury and E.K.P. Chong. "On relative convergence properties of principal component algorithms", IEEE Transactions on Neural Networks, 1998, vol. 9, p. 319–329.

182. Chen H. and R.-W Liu. "Adaptive distributed orthogonalization processing for principal components analysis", International Conference on Acoustics, Speech and Signal Processing, 1992, vol. 2, p. 293–296, San Francisco.

183. Chen S. "Nonlinear time series modelling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning", Electronics Letters, 1995, vol. 31, No. 2, p. 117–118.

184. Chen S., S. Billings and P. Grant. "Non-linear system identification using neural networks", International Journal of Control, 1990, vol. 51, p. 1191–1214.

185. Chen S., B. Mulgrew and S. McLaughlin. "Adaptive Bayesian feedback equalizer based on a radial basis function network", IEEE International Conference on Communications, 1992, vol. 3, p. 1267–1271, Chicago.

186. Cherkassky V. and F. Mulier. "Self-organization as an iterative kernel smoothing process", Neural Computation, 1995, vol. 7, p. 1165–1177.

187. Cherkassky V. and F. Mulier. Learning from Data: Concepts, Theory and Methods, New York: Wiley, 1998.

188. Cherry E.C. "Some experiments on the recognition of speech, with one and with two ears", Journal of the Acoustical Society of America, 1953, vol. 25, p. 975–979.

189. Cherry E.C. and W.K. Taylor. "Some further experiments upon the recognition of speech, with one and with two ears", Journal of Acoustical Society of America, 1954, vol. 26, p. 554–559.

190. Chester D.L. "Why two hidden layers are better than one", International Joint Conference on Neural Networks, 1990, vol. I, p. 265–268, Washington, D.C.

191. Chinrungrueng C. and C.H. Séquin. "Optimal adaptive k-means algorithm with dynamic adjustment of learning rate", IEEE Transactions on Neural Networks, 1994, vol. 6, p. 157–169.

192. Choey M. and A.S. Weigend. "Nonlinear trading models through Sharp ratio maximization", in A. Weigend, Y.S. Abu-Mostafa and A.-P.N. Refenes, eds., 1996, Decision Technologies for Financial Engineering, p. 3–22, Singapore: World Scientific.

193. Churchland P.S. Neurophilosophy: Toward a Unified Science of the Mind/Brain, Cambridge, MA: MIT Press, 1986.

194. Churchland P.S. and T.J. Sejnowski. The Computational Brain, Cambridge, MA: MIT Press, 1992.

195. Cichocki A. and R. Unbehauen. "Robust neural networks with on-line learning for blind identification and blind separation of sources", IEEE Transactions on Circuits and Systems-1: Fundamental Theory and Aplications, 1996, vol. 43, p. 894–906.

196. Cichocki A., R. Ubenhauen and E. Rummert. "Robust learning algorithms for blind separation of signals", Electronics Letters, 1994, vol. 30, p. 1386–1387.

197. Cichocki A. and L. Moszcsynski, L. "New learning algorithm for blind separation of sources", Electronics Letters, 1992, vol. 28, p. 1986–1987.

198. Cleeremans A., D. Servan-Schreiber and J.L. McClelland. "Finite state automata and simple recurrent networks", Neural Computation, 1989, vol. 1, p. 372–381.

199. Cohen L. Time-Frequency Analysis, Englewood Cliffs, NJ: Prentice-Hall, 1995.

200. Cohen M.A. "The synthesis of arbitrary stable dynamics in non-linear neural networks II: Feedback and universality", International Joint Conference on Neural Networks, 1992, vol. I, p. 141–146, Baltimore.

201. Cohen M.A. "The construction of arbitrary stable dynamics in nonlinear neural networks", Neural Networks, 1992Б, vol. 5, p. 83–103.

202. Cohen M.A. and S. Grossberg. "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks", IEEE Transactions on Systems, Man and Cybernetics, 1983, vol. SMC-13, p. 815–826.

203. Cohn D. and G. Tesauro. "How tight are the Vapnik-Chervonenkis bounds"? Neural Computation, 1992, vol. 4, p. 249–269.

204. Comon P. "Supervised classification, a probabilistic aproach", European Symposium on Artificial Neural Networks, 1995, p. 111–128, Brussels, Belgium.

205. Comon P. "Independent component analysis: A new concept"? Signal Processing, 1994, vol. 36, p. 287–314.

206. Comon P. "Independent component analysis", Proceedings of International Signal Processing Workshop on Higher-order Statistics, 1991, p. 111–120, Chamrousse, France.

207. Constantine-Paton M., H.T. Cline and E. Debski. "Patterned activity, synaptic convergence and the NMDA receptor in developing visual pathways", Annual Review of Neuroscience, 1990, vol. 13, p. 129–154.

208. Cook A.S. "The complexity of theorem-proving procedures", Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 1971, p. 151–158, New York.

209. Cook P.A. Nonlinear Dynamical Systems, London: Prentice-Hall International, 1986.

210. Cooper L.N. "A possible organization of animal memory and learning", Proceedings of the Nobel Symposium on Collective Properties of Physical Systems, B. Lundquist and S. Lundquist, eds., 1973, p. 252–264, New York: Academic Press.

211. Cormen T.H., C.E. Leiserson and R.R. Rivest. Introduction to Algorithms. Cambridge, MA: MIT Press, 1990.

212. Cortes C. and V. Vapnik. "Suport vector networks", Machine Learning, 1995, vol. 20, p. 273–297.

213. Cottrell M. and J.C. Fort. "A stochastic model of retinotopy: A self organizing process", Biological Cybernetics, 1986, vol. 53, p. 405–411.

214. Cottrell M., J.C. Fort and G. Pag(s. "Theoretical aspects of the SOM algorithm", Proceedings of the Workshop on Self-Organizing Maps, Espoo, Finland, 1997.

215. Cottrell G.W. and J. Metcalfe. "EMPATH: Face, emotion and gender recognition using holons", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 564–571, San Mateo, CA: Morgan Kaufmann.

216. Cottrell G.W., P. Munro and D. Zipser. "Learning internal representations from grey-scale images: An example of extensional programming", Proceedings of the 9th Annual Conference of the Cognitive science Society, 1987, p. 461–473.

217. Courant R. and D. Hilbert. Methods of Mathematical Physics, 1970, vol. I and II, New York: Wiley Interscience.

218. Cover T.M. "Capacity problems for linear machines", In L. Kanal, ed., Pattern Recognition, 1968, p. 283–289, Washington, DC: Thompson Book Co.

219. Cover T.M. "Geometrical and statistical properties of systems of linear inequalities with aplications in pattern recognition", IEEE Transactions on Electronic Computers, 1965, vol. EC-14, p. 326–334.

220. Cover T.M. and P.E. Hart. "Nearest neighbor pattern classification", IEEE Transactions on Information Theory, 1967, vol. IT-13, p. 21–27.

221. Cover T.M. and J.A. Thomas. Elements of Information Theory, New York: Wiley, 1991.

222. Cowan J.D. "Neural networks: The early days", Advances in Neural Information Processing Systems, 1990, vol. 2, p. 828–842, San Mateo, CA: Morgan Kaufmann.

223. Cowan J.D. "Statistical mechanics of nervous nets", in Neural Networks, E.R. Caianiello, ed., 1968, p. 181–188, Berlin: Springer-Verlag.

224. Cowan J.D. "A Mathematical Theory of Central Nervous Activity", Ph.D. Thesis, University of London, 1967.

225. Cowan J.D. "The problem of organismic reliability", Progress in Brain Research, 1965, vol. 17, p. 9–63.

226. Cowan J.D. and M.H. Cohen. "The role of statistical mechanics in neurobiology", Journal of the Physical Society of Japan, 1969, vol. 26, p. 51–53.

227. Cowan J.D. and D.H. Sharp. "Neural nets", Quarterly Reviews of Biophysics, 1988, vol. 21, p. 365–427.

228. Cragg B.G. and H.N.V. Tamperley. "Memory: The analogy with ferromagnetic hysteresis", Brain, 1955, vol. 78, part II, p. 304–316.

229. Cragg B.G. and H.N.V. Tamperley. "The organization of neurons: A cooperative analogy", EEG Clinical Neurophysiology, 1954, vol. 6, p. 85–92.

230. Craik K.J.W. The Nature of Explanation, Cambridge: Cambridge University Press, 1943.

231. Craven P. and G. Wahba. "Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation", Numerische Mathematik, 1979, vol. 31, p. 377–403.

232. Crick F. "The recent excitement about neural networks", Nature, 1989, vol. 337, p. 129–132.

233. Crites R.H. and A.G. Barto. "Improving elevator performance using reinforcement learning", Advances in Neural Information Processing Systems, 1996, vol. 8, p. 1017–1023, Cambridge, MA: MIT Press.

234. Crutchfield J.P., J.D. Farmer, N.H. Packard and R.S. Shaw. "Chaos", Scientific American, 1986, vol. 255(6), p. 38–49.

235. Cybenko G. "Q-learning: A tutorial and extensions", Presented at Mathematics of Artificial Neural Networks, Oxford University, England, July 1995.

236. Cybenko G. "Aproximation by superpositions of a sigmoidal function", Mathematics of Control, Signals and Systems, 1989, vol. 2, p. 303–314.

237. Cybenko G. "Aproximation by superpositions of a sigmoidal function", Urbana, IL.: University of Illinois, 1988.

238. Darken C. and J. Moody. "Towards faster stochastic gradient search", Advances in Neural Information Processing Systems, 1992, vol. 4, p. 1009–1016, San Mateo, CA: Morgan Kaufmann.

239. Darmois G. "Analyse generale des liaisons stochastiques", Rev. Inst. Internal Stat., 1953, vol. 21, p. 2–8.

240. Dasarathy B.V. ed. Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques, Los Alamitos, CA: IEEE Computer Society Press, 1991.

241. Daubechies I. "The wavelet transform, time-frequency", IEEE Transactions on Information Theory, 1990, vol. IT-36, p. 961–1005.

242. Daubechies I. Ten Lectures on Wavelets, SIAM, 1992.

243. Davis P.J. Interpolation and Aproximation, New York: Blaisdell, 1963.

244. Dayan P. and G.E. Hinton. "Varieties of Helmholtz machine", Neural Networks, 1996, vol. 9, p. 1385–1403.

245. Dayan P., G.E. Hinton, R.M. Neal and R.S. Zemel. "The Helmholtz machine", Neural Computation, 1995, vol. 7, p. 889–904.

246. Debnath L. and P. Mikusiński. Introduction to Hilbert Spaces with Aplications, New York: Academic Press, 1990.

247. Deco G., W. Finnoff and H.G. Zimmermann. "Unsupervised mutual information criteria for elimination of overtraining in supervised multilayer networks", Neural Computation, 1995, vol. 7, p. 86–107.

248. Deco G. and D. Obradovic. An Information-Theoretic Aproach to Neural Computing, New York: Springer, 1996.

249. de Figueiredo R.J.P. "Implications and aplications of Kolmogorov's superposition theorem", IEEE Transactions on Automatic Control, 1980, vol. AC-25, p. 1227–1230.

250. de Figueiredo R.J.P. and G. Chen. Nonlinear Feedback Control Systems, New York: Academic Press, 1993.

251. DeMers D. and G. Cottrell. "Non-linear dimensionality reduction", Advances in Neural Information Processing Systems, 1993, vol. 5, p. 580–587. San Mateo, CA: Morgan Kaufmann.

252. Dempster A.P., N.M. Laird and D.B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm", (with discussion), Journal of the Royal Statistical Society., 1977, B, vol. 39, p. 1–38.

253. Denardo E.V. "Contraction mapings in the theory underlying dynamic programming", SIAM, Review, 1967, vol. 9, p. 165–177.

254. DeSieno, D. "Adding a conscience to competitive learning", IEEE International Conference on Neural Networks, 1988, vol. I, p. 117–124, San Diego, CA.

255. deSilva C.J.S. and Y Attikiouzel. "Hopfield networks as discrete dynamical systems", International Joint Conference on Neural Networks, 1992, vol. III, p. 115–120, Baltimore.

256. deVries B. and J.C. Principe. "The gamma model — A new neural model for temporal processing", Neural Networks, 1992, vol. 5, p. 565–576.

257. Devroye L. "Exponential inequalities in nonparametric estimation", in Nonparametric Functional Estimation and Related Topics, G. Roussas, ed., 1991, p. 31–44. Boston: Kluwer.

258. Diamantaras K.I. and S.Y. Kung. Principal Component Neural Networks: Theory and Aplications, New York: Wiley, 1996.

259. Dohrmann C.R., H.R. Busby and D.M. Trujillo. "Smoothing noisy data using dynamic programming and generalized cross-validation", Journal of Biomechanical Engineering, 1988, vol. 110, p. 37–41.

260. Domany E., J.L. van Hemmen and K. Schulten, eds. Models of Neural Networks, New York: Springer-Verlag, 1991.

261. Dony R.D. and S. Haykin. "Image segmentation using a mixture of principal components representation", IEE Proceedings (London), Image and Signal Processing, 1997, vol. 144, p. 73–80.

262. Dony R.D. and S. Haykin. "Optimally adaptive transform coding", IEEE Transactions on Image Processing, 1995, vol. 4, p. 1358–1370.

263. Dorny C.N. A Vector Space Aproach to Models and Optimization, New York: Wiley (Interscience), 1975.

264. Douglas S.C. and S. Haykin. "On the relationship between blind deconvolution and blind source separation", Thirty-First Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, November, 1997.

265. Doyle J.C., K. Glover, P. Khargonekar and B. Francis. "State-space solutions to standard H_2 and H_∞ control problems", IEEE Transactions on Automatic Control, 1989, vol. AC-34, p. 831–847.

266. Drucker H., C. Cortes, L.D. Jackel and Y. LeCun. "Boosting and other ensemble methods", Neural Computation, 1994, vol. 6, p. 1289–1301.

267. Drucker H., R.E. Schapire and P. Simard. "Improving performance in neural networks using a boosting algorithm", Advances in Neural Information Processing Systems, 1993, vol. 5, p. 42–49, Cambridge, MA: MIT Press.

268. Dubois D. and H. Prade. Fuzzy Sets and Systems: Theory and Aplications, New York: Academic Press, 1980.

269. Duda R.O. and P.E. Hart. Pattern Classification and Scene Analysis, New York: Wiley, 1973.

270. Dunford N. and J.T. Schwartz. Linear Operators, Part 1, New York: Wiley, 1966.

271. Durbin R., C. Miall and G. Mitchison, eds. The Computing Neuron, Reading, MA: Addison-Wesley, 1989.

272. Durbin R. and D.E. Rumelhart. "Product units: A computationally powerful and biologically plausible extension to backpropagation networks", Neural Computation, 1989, vol. 1, p. 133–142.

273. Durbin R. and D. Willshaw. "An analogue aproach to the travelling salesman problem using an elastic net method", Nature, 1987, vol. 326, p. 689–691.

274. Dyn N. "Interpolation of scattered data by radial functions", in Topics in Multivariate Aproximation, C.K. Chui, L.L. Schumaker and F.I. Uteras, eds., 1987, p. 47–61, Orlando, FL: Academic Press.

275. Edelman G.M. Neural Darwinism, New York: Basil Books, 1987.

276. Edelman G.M. "Antibody structure and molecular immunology", Science, 1973, vol. 180, p. 830–840.

277. Eeckman F.H. "The sigmoid nonlinearity in prepyriform cortex", Neural Information Processing Systems, 1988, p. 242–248, New York: American Institute of Physics.

278. Eeckman F.H. and W.J. Freeman. "The sigmoid nonlinearity in neural computation: An experimental aproach", Neural Networks for Computing, J.S. Denker, ed., 1986, p. 135–145, New York: American Institute of Physics.

279. Eggermont J.J. The Correlative Brain: Theory and Experiment in Neural Interaction, New York: Springer-Verlag, 1990.

280. El Hihi S. and Y. Bengio. "Hierarchical recurrent neural networks for long-term dependencies", Advances in Neural Information Processing Systems, 1996, vol. 8, p. 493–499, MIT Press.

281. Elman J.L. "Finding structure in time", Cognitive Science, 1990, vol. 14, p. 179–211.

282. Elman J.L., E.A. Bates, M.H. Johnson, A. Karmiloff-Smith, D. Parisi and K. Plunkett. Rethinking Innateness: A Connectionist Perspective on Development, Cambridge, MA: MIT Press, 1996.

283. Erwin E., K. Obermayer and K. Schulten. "Models of orientation and ocular dominance columns in the visual cortex: A critical comparison", Neural Computation, 1995, vol. 7, p. 425–468.

284. Erwin E., K. Obermayer and K. Schulten. "I: Self-organizing maps: Stationary states, metastability and convergence rate", Biological Cybernetics, 1992, vol. 67, p. 35–45.

285. Erwin E., K. Obermayer and K. Schulten. "II: Self-organizing maps: Ordering, convergence properties and energy functions", Biological Cybernetics, 1992Б, vol. 67, p. 47–55.

286. Faggin F. "VLSI implementation of neural networks", Tutorial Notes, International Joint Conference on Neural Networks, Seattle, 1991.

287. Faggin F. and C. Mead. "VLSI Implementation of Neural Networks", An Introduction to Neural and Electronic Networks, S.F. Zornetzer, J.L. Davis and C. Lau, eds., 1990, p. 275–292, New York: Academic Press.

288. Fahlman S.E. and C. Lebiere. "The cascade-correlation learning architecture", Advances in Neural Information Processing Systems, 1990, vol. 2, p. 524–532, San Mateo, CA: Morgan Kaufmann.

289. Farmer J.D. and J. Sidorowich. "Predicting chaotic time series", Physical Review Letters, 1987, vol. 59, p. 845–848.

290. Feldkamp L.A. and G.V. Puskorius. "Adaptive behavior from fixed weight networks", Information Sciences, 1997, vol. 98, p. 217–235.

291. Feldkamp L.A. and G.V. Puskorius. "A signal processing framework based on dynamic neural networks with aplication to problems in adaptation, filtering and classification", Proceedings of the IEEE, 1998, vol. 86.

292. Feldkamp L.A., G.V. Puskorius and P.C. Moore. "Adaptation from fixed weight networks", Information Sciences, 1997, vol. 98, p. 217–235.

293. Feldraan J.A. "Natural computation and artificial intelligence", Plenary Lecture presented at the International Joint Conference on Neural Networks, Baltimore, 1992.

294. Feller W., 1968. An Introduction to Probability Theory and its Aplications, vol. 1, 3rd edition, New York: John Wiley; 1st edition, 1950.

295. Fischler M.A. and O. Firschein. Intelligence: The Eye, The Brain and The Computer, Reading, MA: Addison-Wesley, 1987.

296. Fisher R.A. "Theory of statistical estimation", Proceedings of the Cambridge Philosophical Society, 1925, vol. 22, p. 700–725.

297. Fix E. and J.L. Hodges. "Discriminatory analysis: Nonparametric discrimination: Consistency properties", USAF School of Aviation Medicine, 1951, Project 21-49-004, Report no. 4, p. 261–279, Randolph Field, Texas.

298. Fletcher R. Practical Methods of Optimization, 2nd edition, New York: Wiley, 1987.

299. Fodor J.A. Modularity of Mind, Cambridge, MA: MIT Press, 1983.

300. Fodor J.A. and Z.W. Pylyshyn. "Connectionism and cognitive architecture: a critical analysis", Cognition, 1988, vol. 28, p. 3–72.

301. Foldiak P. "Adaptive network for optimal linear feature extractions", IEEE International Joint Conference on Neural Networks, 1989, vol. I, p. 401–405, Washington, DC.

302. Forcada M.L. and R.C. Carrasco. "Learning the initial state of a second-order recurrent neural network during regular-language inference", Neural Computation, 1995, vol. 7, p. 923–930.

303. Forney G.D., Jr. "The Viterbi algorithm", Proceedings of the IEEE, 1973, vol. 61, p. 268–278.

304. Forte J.C. and G. Pagés. "On the a.s. convergence of the Kohonen algorithm with a general neighborhood function", Annals of Aplied Probability, 1995, vol. 5, p. 1177–1216.

305. Forte J.C. and G. Pagés. "Convergence of stochastic algorithm: From the Kushner and Clark theorem to the Lyapunov functional", Advances in Aplied Probability, 1996, vol. 28, p. 1072–1094.

306. Frasconi P., M. Gori and G. Soda. "Local feedback multilayered networks", Neural Computation, 1992, vol. 4, p. 120–130.

307. Frasconi P. and M. Gori. "Computational capabilities of local-feedback recurrent networks acting as finite-state machines", IEEE Transactions on Neural Networks, 1996, vol. 7, p. 1521–1524.

308. Fraser A.M. "Information and entropy in strange attractors", IEEE Transactions on Information Theory, 1989, vol. 35, p. 245–262.

309. Freeman J.A. and D.M. Sakpura. Neural Networks: Algorithms, Aplications and Programming Techniques, Reading, MA: Addison-Wesley, 1991.

310. Freeman W.J. Societies of Brains. Hillsdale, NJ: Lawrence Erlbaum, 1995.

311. Freeman W.J. "Tutorial on neurobiology: From single neurons to brain chaos", International Journal of Bifurcation and Chaos in Aplied Sciences and Engineering, 1992, vol. 2, p. 451–482.

312. Freeman W.J. "The physiology of perception", Scientific American, 1991, vol. 264 (2), p. 78–85.

313. Freeman W.J. "Why neural networks don't yet fly: Inquiry into the neurodynamics of biological intelligence", IEEE International Conference on Neural Networks, 1988, vol. II, p. 1–7, San Diego, CA.

314. Freeman W.J. "Simulation of chaotic EEG patterns with a dynamic model of the olfactory system", Biological Cybernetics, 1987, vol. 56, p. 139–150.

315. Freeman W.J. Mass Action in the Nervous System, New York: Academic Press, 1975.

316. Frégnac Y. and D. Schulz. "Models of synaptic plasticity and cellular analogs of learning in the developing and adult vertebrate visual cortex", Advances in Neural and Behavioral Development, 1994, vol. 4, p. 149–235, Norwood, NJ: Neural Ablex.

317. Freund Y. "Boosting a week learning algorithm by majority", Information Computation, 1995, vol. 121, p. 256–285.

318. Freund Y. and R.E. Schapire. "A decision-theoretic generalization of on-line learning and an aplication to boosting", Journal of Computer and System Sciences, 1997, vol. 55, p. 119–139.

319. Freund Y. and R.E. Schapire. "Experiments with a new boosting algorithm", Machine Learning: Proceedings of the Thirteenth International Conference, 1996, p. 148–156, Bari, Italy.

320. Freund Y. and R.E. Schapire. "Game theory, On-line prediction and boosting", Proceedings of the Ninth Annual Conference on Computational Learning Theory, 1996Б, p. 325–332, Desenzano del Garda, Italy.

321. Friedman J.H. "An overview of prediction learning and function aproximation", In V. Cherkassky, J.H. Friedman and H. Wechsler, eds., From Statistics to Neural Networks: Theory and Pattern Recognition Aplications, New York: Springer-Verlag, 1995.

322. Fukunaga K. Statistical Pattern Recognition, 2nd edition, New York: Academic Press, 1990.

323. Fukushima K. "Neocognitron: A model for visual pattern recognition", in M.A. Arbib, ed., The Handbook of Brain Theory and Neural Networks, Cambridge, MA: MIT Press, 1995.

324. Fukushima K. "A hierarchical neural network model for selective attention", in Neural Computers, R. Eckmiller and C. von der Malsburg, eds., 1988, p. 81–90, NATO ASI Series, New York: Springer-Verlag.

325. Fukushima K. "Neocognitron: A hierarchical neural network capable of visual pattern recognition", Neural Networks, 1988Б, vol. l, p. 119–130.

326. Fukushima K. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", Biological Cybernetics, 1980, vol. 36, 193–202.

327. Fukushima K. "Cognitron: A self-organizing multi-layered neural network", Biological Cybernetics, 1975, vol. 20, p. 121–136.

328. Fukushima K., S. Miyake and T. Ito. "Neocognitron: A neural network model for a mechanism of visual pattern recognition", IEEE Transactions on Systems, Man and Cybernetics, 1983, vol. SMC-13, p. 826–834.

329. Funahashi K. "On the aproximate realization of continuous mapings by neural networks", Neural Networks, 1989, vol. 2, p. 183–192.

330. Gabor D. "Communication theory and cybernetics", IRE Transactions on Circuit Theory, 1954, vol. CT-1, p. 19–31.

331. Gabor D., W.P.L. Wilby and R. Woodcock. "A universal non-linear filter, predictor and simulator which optimizes itself by a learning process", Proceedings of the Institution of Electrical Engineers, London, 1960, vol. 108, p. 422–435.

332. Galland C.C. "The limitations of deterministic Boltzmann machine learning", Network, 1993, vol. 4, p. 355–379.

333. Gallant A.R. and H. White. "There exists a neural network that does not make avoidable mistakes", IEEE International Conference on Neural Networks, 1988, vol. I, p. 657–664, San Diego, CA.

334. Gallant A.R. and H. White. "On learning the derivatives of an unknown maping with multilayer feedforward networks", Neural Networks, 1992, vol. 5, p. 129–138.

335. Gallant S.I. Neural Network Learning and Expert Systems, Cambridge, MA: MIT Press, 1993.

336. Gallistel C.R. The Organization of Learning, Cambridge, MA: MIT Press, 1990.

337. Gardner E. "Maximum storage capacity in neural networks", Electrophysics Letters, 1987, vol. 4, p. 481–485.

338. Garey M.R. and D.S. Johnson. Computers and Intractability, New York: W.H. Freeman, 1979.

339. Gee A.H. "Problem solving with optimization networks", Ph.D. dissertation, University of Cambridge, 1993.

340. Gee A.H., S.V.B. Aiyer and R. Prager. "An analytical framework for optimizing neural networks", Neural Networks, 1993, vol. 6, p. 79–97.

341. Geisser S. "The predictive sample reuse method with aplications", Journal of the American Statistical Association, 1975, vol. 70, p. 320–328.

342. Gelfand A.E. and A.F.M. Smith. "Sampling-based aproaches to calculating marginal densities", Journal of the American Statistical Association, 1990, vol. 85, p. 398–409.

343. Geman S. and D. Geman. "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1984, vol. PAMI-6, p. 721–741.

344. Geman S., E. Bienenstock and R. Doursat. "Neural networks and the bias/variance dilemma", Neural Computation, 1992, vol. 4, p. 1–58.

345. Gersho A. "On the structure of vector quantizers", IEEE Transactions on Information Theory, 1982, vol. IT-28, p. 157–166.

346. Gersho A. and R.M. Gray. Vector Quantization and Signal Compression, Norwell, MA: Kluwer, 1992.

347. Gerstein G.L., P. Bedenbaugh and A.M.H.J. Aersten. "Neural assemblies", IEEE Transactions on Biomedical Engineering, 1989, vol. 36, p. 4–14.

348. Gibbs J.W., 1902. "Elementary principles in statistical mechanics", reproduced in vol. 2 of Collected Works of J. Willard Gibbs in Two Volumes, New York: Longmans, Green and Co., 1928.

349. Gibson G.J. and C.F.N. Cowan. "On the decision regions of multilayer perceptrons", Proceedings of the IEEE, 1990, vol. 78, p. 1590–1599.

350. Gidas B. "Global optimization via the Langevin equation", Proceedings of 24th Conference on Decision and Control, 1985, p. 774–778, Ft. Lauderdale, FL.

351. Giles C.L. "Dynamically driven recurrent neural networks: Models, learning algorithms and aplications", Tutorial #4, International Conference on Neural Networks, Washington, DC, 1996.

352. Giles C.L., D. Chen, G.Z. Sun, H.H. Chen, Y.C. Lee and M.W. Goudreau. "Constructive learning of recurrent neural networks: Limitations of recurrent cascade correlation with a simple solution", IEEE Transactions on Neural Networks, 1995, vol. 6, p. 829–836.

353. Giles C.L., T. Lin and B.G. Horne. "Remembering the past: The role of embedded memory in recurrent neural network architectures", Neural Networks for Signal Processing, VII, Proceedings of the 1997 IEEE Workshop, IEEE Press, 1997, p. 34.

354. Giles C.L. and T. Maxwell. "Learning, invariance and generalization in higher-order neural networks", Aplied Optics, 1987, vol. 26, p. 4972–4978.

355. Giles C.L. and B.G. Horne. "Representation of learning in recurrent neural network architectures", Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems, 1994, p. 128–134, Yale University, New Haven, Ct.

356. Giles C.L., C.B. Miller D. Chen, H.H. Chen, G.Z. Sun and Y.C. Lee. "Learning and extracting finite state automata with second-order recurrent neural networks", Neural Computation, 1992, vol. 4, p. 393–405.

357. Giles C.L., G.Z. Sun, H.H. Chen, Y.C. Lee and D. Chen. "Higher order recurrent networks and grammatical inference", Advances in Neural Information Processing Systems, 1990, vol. 2, p. 380–387, San Mateo, CA: Morgan Kaufmann.

358. Gill P., S. Hammarling, W. Murray, M. Saunders and M. Wright. "User's guide for LSSOL", Technical Report 86-1, Systems Optimization Laboratory, Stanford University, Stanford, CA, 1986.

359. Gill P. and W. Murray. "Inertia-controlling methods for general quadratic programming", SIAM Review, 1991, vol. 33, p. 1–36.

360. Girosi F. and G. Anzellotti. "Rates of convergence of aproximation by translates", A.I. Memo 1288, Artificial Intelligence Laboratory, MIT Cambridge, MA, 1992.

361. Girosi F., M. Jones and T. Poggio. "Regularization theory and neural networks architectures", Neural Computation, 1995, vol. 7, p. 219–269.

362. Girosi F. and T. Poggio. "Networks and the best aproximation property", Biological Cybernetics, 1990, vol. 63, p. 169–176.

363. Glauber R.J. "Time-dependent statistics of the Ising model", Journal of Mathematical Physics, 1963, vol. 4, p. 294–307.

364. Goggin S.D.D., K.M. Johnson and K. Gustafson. "Primary and recency effects due to momentum in back-propagation learning", OCS Technical Report 89-25, Boulder, CO.: University of Colorado, 1989.

365. Golden R.M. Mathematical Methods for Neural Network Analysis and Design, Cambridge, MA: MIT Press, 1996.

366. Golden R.M. "The ?Brain-State-in-a-Box? neural model is a gradient descent algorithm", Journal of Mathematical Psychology, 1986, vol. 30, p. 73–80.

367. Goles E. and S. Martinez. Neural and Automata Networks, Dondrecht, The Netherlands: Kluwer, 1990.

368. Golub G.H. and C.G. Van Loan. Matrix Computations, 3rd edition, Baltimore: Johns Hopkins University Press, 1996.

369. Goodman R.M., C.M. Higgins, J.W. Miller and P. Smyth. "Rule-based neural networks for classification and probability estimation", Neural Computation, 1992, vol. 4, p. 781–804.

370. Gori M. and A. Tesi. "On the problem of local minima in backpropagation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992, vol. 14, p. 76–86.

371. Gorin A. "Network structure, generalization and adaptive language acquisition", Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems, 1992, p. 155–160, Yale University, New Haven, CT.

372. Goudreau M.W. and C.L. Giles. "Using recurrent neural networks to learn the structure of interconnection networks", Neural Networks, 1995, vol. 8, p. 793–804.

373. Goudreau M.W., C.L. Giles, S.T. Chakradhar and D. Chen. "First-order vs. second-order single-layer recurrent neural networks", IEEE Transactions on Neural Networks, 1994, vol. 5, p. 511–513.

374. Granger R., J. Whitson, J. Larson and G. Lynch. "Non-Hebbian properties of LTP enable high-capacity encoding of temporal sequences", Proceedings of the National Academy of Sciences of the U.S.A, 1994.

375. Grassberger I. and I. Procaccia. "Measuring the strangeness of strange attractors", Physica D, 1983, vol. 9, p. 189–208.

376. Graubard S.R., ed. The Artificial Intelligence Debate: False Starts, Real Foundations, Cambridge, MA: MIT Press, 1988.

377. Gray R.M. Entropy and Information Theory, New York: Springer-Verlag, 1990.

378. Gray R.M. Probability, Random Processes and Ergodic Properties, New York: Springer-Verlag, 1988.

379. Gray R.M. "Vector quantization", IEEE ASSP Magazine, 1984, vol. 1, p. 4–29.

380. Gray R.M. and L.D. Davisson. Random Processes: A Mathematical Aproach for Engineers, Englewood Cliffs, NJ: Prentice-Hall, 1986.

381. Green M. and D.J.N. Limebeer. Linear Robust Control, Englewood Cliffs, NJ: Prentice-Hall, 1995.

382. Greenberg H.J. "Equilibria of the brain-state-in-a-box (BSB) neural model", Neural Networks, 1988, vol. 1, p. 323–324.

383. Gregory R.L. The Intelligent Eye, Wiedefeld and Nicholson, London, 1970.

384. Grenander U. Tutorial in Pattern Theory, Brown University, Providence, R.I., 1983.

385. Grewal M.S. and A.P. Andrews. Kalman Filtering: Theory and Practice, Englewood Cliffs, NJ: Prentice-Hall, 1993.

386. Griffiths L.J. and C.W. Jim. "An alternative aproach to linearly constrained optimum beamforming", IEEE Transactions on Antennas and Propagation, 1982, vol. AP-30, p. 27–34.

387. Grossberg S. "Content-addressable memory storage by neural networks: A general model and global Liapunov method", In Computational Neuroscience, E.L. Schwartz, ed., 1990, p. 56–65, Cambridge, MA: MIT Press.

388. Grossberg S. "Competitive learning: From interactive activation to adaptive resonance", in Neural Networks and Natural Intelligence, S. Grossberg, ed., Cambridge, MA: MIT Press, 1988.

389. Grossberg S.Z. Neural Networks and Natural Intelligence, Cambridge, MA: MIT Press, 1988Б.

390. Grossberg S. "Nonlinear neural networks: Principles, mechanisms and architectures", Neural Networks, 1988б, vol. 1, p. 17–61.

391. Grossberg S. Studies of Mind and Brain, Boston: Reidel, 1982.

392. Grossberg S. "How does a brain build a cognitive code"? Psychological Review, 1980, vol. 87, p. 1–51.

393. Grossberg S. "Decision, patterns and oscillations in the dynamics of competitive systems with aplication to Volterra-Lotka systems", J. Theoretical Biology, 1978, vol. 73, p. 101–130.

394. Grossberg S. "Competition, decision and consensus", J. Mathematical Analysis and Aplications, 1978Б, vol. 66, p. 470–493.

395. Grossberg S. "Pattern formation by the global limits of a nonlinear competitive interaction in n dimensions", J. Mathematical Biology, 1977, vol. 4, p. 237–256.

396. Grossberg S. "Adaptive pattern classification and universal receding: I. Parallel development and coding of neural detectors", Biological Cybernetics, 1976, vol. 23, p. 121–134.

397. Grossberg S. "Adaptive pattern classification and universal receding: II. Feedback, expectation, olfaction, illusions", Biological Cybernetics, 1976Б, vol. 23, p. 187–202.

398. Grossberg S. "Neural expectation: Cerebellar and retinal analogs of cells fired by learnable or unlearned pattern classes", Kybernetik, 1972, vol. 10, p. 49–57.

399. Grossberg S. "A prediction theory for some nonlinear functional-difference equations", Journal of Mathematical Analysis and Aplications, 1969, vol. 22, p. 490–522.

400. Grossberg S. "On learning and energy-entropy dependence in recurrent and nonrecurrent signed networks", Journal of Statistical Physics, 1969Б, vol. 1, p. 319–350.

401. Grossberg S. "A prediction theory for some nonlinear functional-difference equations", Journal of Mathematical Analysis and Aplications, 1968, vol. 21, p. 643–694, vol. 22, p. 490–522.

402. Grossberg S. "Nonlinear difference – differential equations in prediction and learning theory", Proceedings of the National Academy of Sciences, 1967, USA, vol. 58, p. 1329–1334.

403. Gupta M.M. and N.K. Sinha, eds. Intelligent Control Systems: Theory and Aplications, New York: IEEE Press, 1996.

404. Guyon I. Neural Networks and Aplications, Computer Physics Reports, Amsterdam: Elsevier, 1990.

405. Haffner P. "A new probabilistic framework for connectionist time alignment", Proceedings of ICSLP 94, 1994, p. 1559–1562, Yokohama, Japan.

406. Haffner P., M. Franzini and A. Waibel. "Integrating time alignment and neural networks for high performance continuous speech recognition", Proceedings of IEEE ICASSP 91, 1991, p. 105–108.

407. Haft M. and J.L. van Hemmen. "Theory and implementations of infomax filters for the retina", Network: Computations in Neural Systems, 1998, vol. 9, p. 39–71.

408. Hagiwara M. "Theoretical derivation of momentum term in back-propagation", International Joint Conference on Neural Networks, 1992, vol. I, p. 682–686, Baltimore.

409. Hajek B. "A tutorial survey of theory and aplications of simulated annealing", Proceedings of the 24th Conference on Decision and Control, IEEE Press, 1985, p. 755–760, Ft. Lauderdale, Fla.

410. Hajek B. "Cooling schedules for optimal annealing", Mathematics of Operations Research, 1988, vol. 13, p. 311–329.

411. Hammerstrom D. "Neural networks at work", IEEE Spectrum, 1993, vol. 30, no. 6, p. 26–32.

412. Hammerstrom D. "Working with neural networks", IEEE Spectrum, 1993Б, vol. 30, no. 7, p. 46–53.

413. Hammerstrom D. and S. Rahfuss. "Neurocomputing hardware: Present and future", Swedish National Conference on Connectionism, Skovade, Sweden, September, 1992.

414. Hampshire J.B. and B. Pearlmutter. "Equivalence proofs for multilayer perceptron classifiers and Bayesian discriminant function", Proceedings of the 1990 Connectionist Models Summer School, 1990, p. 159–172, San Mateo, CA: Morgan Kaufmann.

415. Hampson S.E. Connectionistic Problem Solving: Computational Aspects of Biological Learning, Berlin: Birkh(user, 1990.

416. Hancock P.J.B., R.J. Baddeley and L.S. Smith. "The principal components of natural images", Network, 1992, vol. 3, p. 61–70.

417. Hanson L.K. and P. Solamon. "Neural network ensembles", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, vol. PAMI-12, p. 993–1002.

418. Härdle W. Aplied Nonparametric Regression, Cambridge: Cambridge University Press, 1990.

419. Hardy R.L. "Multiquadric equations of topography and other irregular surfaces", Journal of Geophysics Research, 1971, vol. 76, p. 1905–1915.

420. Harel D. "Algorithmics: The Spirit of Computing", Reading, MA: Addison-Wesley, 1987.

421. Hartline H.K. "The receptive fields of optic nerve fibers", American Journal of Physiology, 1940, vol. 130, p. 690–699.

422. Hartman E. "A high storage capacity neural network content-addressable memory", Network, 1991, vol. 2, p. 315–334.

423. Hartman, E.J., J.D. Keeler and J.M. Kowalski. "Layered neural networks with Gaussian hidden units as universal aproximators", Neural Computation, 1990, vol. 2, p. 210–215.

424. Hashem S. "Optimal linear combinations of neural networks", Neural Networks, 1997, vol. 10, p. 599–614.

425. Hassibi B., A.H. Sayed and T. Kailath. Indefinite Quadratic Estimation and Control: A Unified Aproach to H2 and H? Theories, SIAM, 1998.

426. Hassibi B., A.H. Sayed and T. Kailath. "The H^{∞} optimality of the LMS algorithm", IEEE Transactions on Signal Processing, 1996, vol. 44, p. 267–280.

427. Hassibi B., A.H. Sayed and T. Kailath. "LMS is H^{∞} optimal", Proceedings of the IEEE Conference on Decision and Control, 1993, p. 74–79, San Antonio, Texas.

428. Hassibi B., D.G. Stork and G.J. Wolff. "Optimal brain surgeon and general network pruning", IEEE International Conference on Neural Networks, 1992, vol. 1, p. 293–299, San Francisco.

429. Hassibi B. and T. Kailath. "H? optimal training algorithms and their relation to back propagation", Advances in Neural Information Proccessing Systems, 1995, vol. 7, p. 191–198.

430. Hastie T. and W. Stuetzle. "Principal curves", Journal of the American Statistical Association, 1989, vol. 84, p. 502–516.

431. Hastings W.K. "Monte Carlo sampling methods using Markov chains and their aplications", Biometrika, 1970, vol. 87, p. 97–109.

432. Haussler D. "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework", Artificial Intelligence, 1988, vol. 36, p. 177–221.

433. Hawkins R.D. and G.H. Bower, eds. Computational Models of Learning in Simple Neural Systems, San Diego, CA: Academic Press, 1989.

434. Haykin S. Adaptive Filter Theory, 3rd edition, Englewood Cliffs, NJ: Prentice-Hall, 1996.

435. Haykin S. "Neural networks expand SP's horizons", IEEE Signal Processing Magazine, 1996Б, vol. 13, no. 2, p. 24–29.

436. Haykin S., ed. Blind Deconvolution, Englewood Cliffs, NJ: Prentice-Hall, 1994.

437. Haykin S. Communication Systems, 3rd edition, New York: John Wiley, 1994Б.

438. Haykin S. "Blind equalization formulated as a self-organized learning process", Proceedings of the Twenty-Sixth Asilomar Conference on Signals, Systems and Computers, 1992, p. 346–350, Pacific Grove, CA.

439. Haykin S. and C. Deng. "Classification of radar clutter using neural networks", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 589–600.

440. Haykin S. and B. Kosko, eds. Special Issue of Proceedings of the IEEE on Intelligent Signal Processing, 1998, vol. 88.

441. Haykin S. and J. Principe. "Making sense of a complex world: Using neural networks to dynamically model chaotic events such as sea clutter", IEEE Signal Processing Magazine, 1998, vol. 15.

442. Haykin S., W. Stehwien, P. Weber, C. Deng and R. Mann. "Classification of radar clutter in air traffic control environment", Proceedings of the IEEE, 1991, vol. 79, p. 741–772.

443. Haykin S., P. Yee and E. Derbez. "Optimum nonlinear filtering", IEEE Transactions on Signal Processing, 1997, vol. 45, p. 2774–2786.

444. Haykin S. and B. Van Veen. Signals and Systems, New York: Wiley, 1998.

445. Hebb D.O. The Organization of Behavior: A Neuropsychological Theory, New York: Wiley, 1949.

446. Hecht-Nielsen R. "Replicator neural networks for universal optimal source coding", Science, 1995, vol. 269, p. 1860–1863.

447. Hecht-Nielson R., 1990. Neurocomputing, Reading, MA: Addison-Wesley.

448. Hecht-Nielson R. "Kolmogorov's maping neural network existence theorem", First IEEE International Conference on Neural Networks, 1987, vol. III, p. 11–14, San Diego, CA.

449. Helstrom C.W. Statistical Theory of Signal Detection, 2nd edition, Pergamon Press, 1968.

450. Herault J. and C. Jutten. Reseaux Neuronaux et Traitement du Signal, Paris: Hermes Publishers, 1994.

451. Herault J. and C. Jutten. "Space or time adaptive signal processing by neural network models", in J.S. Denker, ed., Neural Networks for Computing. Proceedings of the AIP Conference, American Institute of Physics, New York, 1986, p. 206–211.

452. Herault J., C. Jutten and B. Ans. "Detection de grandeurs primitives dans un message composite par une architecture de calcul neuromimetique un aprentissage non supervise", Procedures of GRETSI, Nice, France, 1985.

453. Hertz J., A. Krogh and R.G. Palmer. Introduction to the Theory of Neural Computation, Reading, MA: Addison-Wesley, 1991.

454. Hestenes M.R. and E. Stiefel. "Methods of conjugate gradients for solving linear systems", Journal of Research of the National Bureau of Standards, 1952, vol. 49, p. 409–436.

455. Hetherington P.A. and M.L. Shapiro. "Simulating Hebb cell assemblies: The necessity for partitioned dendritic trees and a post-not-pre LTD rule", Network, 1993, vol. 4, p. 135–153.

456. Hiller F.S. and G.J. Lieberman. Introduction to Operations Research, 6th edition, New York: McGraw-Hill, 1995.

457. Hinton G.E. "Connectionist learning procedures", Artificial Intelligence, 1989, vol. 40, p. 185–234.

458. Hinton G.E. "Deterministic Boltzmann machine learning performs steepest descent in weight-space", Neural Computation, 1989, vol. l, p. 143–150.

459. Hinton G.E. "Shape representation in parallel systems", Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver, British Columbia, 1981.

460. Hinton G.E., P. Dayan, B.J. Frey and R.M. Neal. "The 'wake-sleep' algorithm for unsupervised neural networks", Science, 1995, vol. 268, p. 1158–1161.

461. Hinton G.E. and Z. Ghahramani. "Generative models for discovering sparse distributed representations", Philosophical Transactions of the Royal Society, Series B, 1997, vol. 352, p. 1177–1190.

462. Hinton G.E. and S.J. Nowlan. "The bootstrap Widrow-Hoff rule as a cluster-formation algorithm", Neural Computation, 1990, vol. 2, p. 355–362.

463. Hinton G.E. and S.J. Nowlan. "How learning can guide evolution", Complex Systems, 1987, vol. 1, p. 495–502.

464. Hinton G.E. and T.J. Sejnowski. "Learning and relearning in Boltzmann machines", in Parallel Distributed Processing: Explorations in Microstructure of Cognition, D.E. Rumelhart and J.L. McClelland, eds., Cambridge, MA: MIT Press, 1986.

465. Hinton G.E. and T.J. Sejnowski. "Optimal perceptual inference", Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1983, p. 448–453, Washington, DC.

466. Hirsch M.W. "Convergent activation dynamics in continuous time networks", Neural Networks, 1989, vol. 2, p. 331–349.

467. Hirsch M.W. "Convergence in neural nets", First IEEE International Conference on Neural Networks, 1987, vol. II, p. 115–125, San Diego, CA.

468. Hirsch M.W. and S. Smale. Differential Equations, Dynamical Systems and Linear Algebra, New York: Academic Press, 1974.

469. Hochreiter S. Untersuchungen zu dynamischen neuronalen Netzen, Diploma Thesis, Technische Universität München, Germany, 1991.

470. Hochreiter S. and J. Schmidhuber. "LSTM can solve hard long time lag problems", Advances in Neural Information Processing Systems, 1997, vol. 9, p. 473–479, Cambridge, MA: MIT Press.

471. Hodgkin A.L. and A.F. Huxley. "A quantitative description of membrane current and its aplication to conduction and excitation in nerve", Journal of Physiology, 1952, vol. 117, p. 500–544.

472. Holden S.B. and M. Niranjan. "On the practical aplicability of Vapnik-Chervonenkis dimension bounds", Neural Computation, 1995, vol. 7, p. 1265–1288.

473. Holland J.H. Adaptation in Natural and Artificial Systems, Cambridge, MA: MIT Press, 1992.

474. Hopcroft J. and U. Ullman. Introduction to Automata Theory, Languages and Computation, Reading, MA:Addison-Wesley, 1979.

475. Hopfield J.J. "Pattern recognition computation using action potential timing for stimulus representation", Nature, 1995, vol. 376, p. 33–36.

476. Hopfield J.J. "Neurons, dynamics and computation", Physics Today, 1994, vol. 47, p. 40–46, February.

477. Hopfield J.J. "Networks, Computations, Logic and Noise", IEEE International Conference on Neural Networks, 1987, vol. I, p. 107–141, San Diego, CA.

478. Hopfield J.J. "Learning algorithms and probability distributions in feed-forward and feed-back networks", Proceedings of the National Academy of Sciences, USA, 1987Б, vol. 84, p. 8429–8433.

479. Hopfield J.J. "Neurons with graded response have collective computational properties like those of two-state neurons", Proceedings of the National Academy of Sciences, USA, 1984, vol. 81, p. 3088–3092.

480. Hopfield J.J. "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences, USA, 1982, vol. 79, p. 2554–2558.

481. Hopfield J.J. and D.W. Tank. "Computing with neural circuits: A model", Science, 1986, vol. 233, p. 625–633.

482. Hopfield J.J. and T.W. Tank. "'Neural' computation of decisions in optimization problems", Biological Cybernetics, 1985, vol. 52, p. 141–152.

483. Hopfield J.J., D.I. Feinstein and R.G. Palmer. "'Unlearning' has a stabilizing effect in collective memories", Nature, 1983, vol. 304, p. 158–159.

484. Horn B.K.P. "Understanding image intensities", Artificial Intelligence, 1977, vol. 8, p. 201–237.

485. Hornik K., M. Stinchcombe and H. White. "Universal aproximation of an unknown maping and its derivatives using multilayer feedforward networks", Neural Networks, 1990, vol. 3, p. 551–560.

486. Hornik K., M. Stinchcombe and H. White. "Multilayer feedforward networks are universal aproximators", Neural Networks, 1989, vol. 2, p. 359–366.

487. Hotteling H. "Analysis of a complex of statistical variables into principal components", Journal of Educational Psychology, 1933, vol. 24, p. 417–441, 498–520.

488. Hubel D.H. Eye, Brain and Vision, New York: Scientific American Library, 1988.

489. Hubel D.H. and T.N. Wiesel. "Functional architecture of macaque visual cortex", Proceedings of the Royal Society, B, 1977, vol. 198, p. 1–59, London.

490. Hubel D.H. and T.N. Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex", Journal of Physiology, 1962, vol. 160, p. 106–154, London.

491. Huber P.J. "Projection pursuit", Annals of Statistics, 1985, vol. 13, p. 435–475.

492. Huber P.J. Robust Statistics, New York: Wiley, 1981.

493. Huber P.J. "Robust estimation of a location parameter", Annals of Mathematical Statistics, 1964, vol. 35, p. 73–101.

494. Hush D.R. "Learning from examples: From theory to practice", Tutorial #4, 1997 International Conference on Neural Networks, Houston, June, 1997.

495. Hush D.R. and B.G. Horne. "Progress in supervised neural networks: What's new since Lipmann"? IEEE Signal Processing Magazine, 1993, vol. 10, p. 8–39.

496. Hush D.R. and J.M. Salas. "Improving the learning rate of back-propagation with the gradient reuse algorithm", IEEE International Conference on Neural Networks, 1988, vol. I, p. 441–447, San Diego, CA.

497. Illingsworth V., E.L. Glaser and I.C. Pyle. Dictionary of Computing, New York: Oxford University Press, 1989.

498. Intrator N. "Feature extraction using an unsupervised neural network", Neural Computation, 1992, vol. 4, p. 98–107.

499. Jaakkola T. and M.I. Jordan. "Computing uper and lower bounds on likelihoods in intractable networks", in E. Horwitz, ed., Workshop on Uncertainty in Artificial Intelligence, Portland, Or, 1996.

500. Jackson E.A. Perspectives of Nonlinear Dynamics, vol. 1, Cambridge: Cambridge University Press, 1989.

501. Jackson E.A. Perspectives of Nonlinear Dynamics, vol. 2, Cambridge: Cambridge University Press, 1990.

502. Jackson I.R.H. "An order of convergence for some radial basis functions", IMA Journal of Numerical Analysis, 1989, vol. 9, p. 567–587.

503. Jackson J.D. Classical Electrodynamics, 2nd edition, New York: Wiley, 1975.

504. Jacobs R.A. "Task Decomposition Through Computation in a Modular Connectionist Architecture", Ph.D. Thesis, University of Massachusetts, 1990.

505. Jacobs R.A. "Increased rates of convergence through learning rate adaptation", Neural Networks, 1988, vol. 1, p. 295–307.

506. Jacobs R.A. and M.I. Jordan. "Learning piecewise control strategies in a modular neural network architecture", IEEE Transactions on Systems, Man and Cybernetics, 1993 vol. 23, p. 337–345.

507. Jacobs R.A. and M.I. Jordan. "A competitive modular connectionist architecture", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 767–773, San Mateo, CA: Morgan Kaufmann.

508. Jacobs R.A., M.I. Jordan, S.J. Nowlan and G.E. Hinton. "Adaptive mixtures of local experts", Neural Computation, 1991, vol. 3, p. 79–87.

509. Jacobs R.A., M.I. Jordan and A.G. Barto. "Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks", Cognitive Science, 1991Б, vol. 15, p. 219–250.

510. Jayant N.S. and P. Noll. Digital Coding of Waveforms, Englewood Cliffs, NJ: Prentice-Hall, 1984.

511. Jaynes E.T. "On the rationale of maximum-entropy methods", Proceedings of the IEEE, 1982, vol. 70, p. 939–952.

512. Jaynes E.T. "Information theory and statistical mechanics", Physical Review, vol. 106, p. 620–630; "Information theory and statistical mechanic II", Physical Review, 1957, vol. 108, p. 171–190.

513. Jazwinski A.H. Stochastic Processes and Filtering Theory, New York: Academic Press, 1970.

514. Jelinek E. Statistical Methods for Speech Recognition, Cambridge, MA: MIT Press, 1997.

515. Johansson E.M., F.U. Dowla and D.M. Goodman. "Back-propagation learning for multi-layer feedforward neural networks using the conjugate gradient method", Report UCRL-JC-104850, Lawrence Livermore National Laboratory, CA, 1990.

516. Johnson D.S., C.R. Aragon, L.A. McGeoch and C. Schevon. "Optimization by simulated annealing: An experimental evaluation", Operations Research, 1989, vol. 37, p. 865–892.

517. Jolliffe I.T. Principal Component Analysis, New York: Springer-Verlag, 1986.

518. Jones J.P. and L.A. Palmer. "The two-dimensional spatial structure of simple receptive fields in cat striate cortex", Journal of Neurophysiology, 1987, vol. 58, p. 1187–1211.

519. Jones J.P. and L.A. Palmer. "An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex", Journal of Neurophysiology, 1987Б, vol. 58, p. 1233–1258.

520. Jones J.P., A. Steponski and L.A. Palmer. "The two-dimensional spectral structure of simple receptive fields in cat striate cortex", Journal of Neurophysiology, 1987, vol. 58, p. 1212–1232.

521. Jordan M.I. "A statistical aproach to decision tree modeling", Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory, New York: ACM Press, 1994.

522. Jordan M.I. "Attractor dynamics and parallelism in a connectionist sequential machine", The Eighth Annual Conference of the Cognitive Science Society, 1986, p. 531–546, Amherst, MA.

523. Jordan M.I., ed. Learning in Graphical Models, Boston: Kluwer, 1998.

524. Jordan M.I., Z. Ghahramani, T.S. Jakkolla and L.K. Saul. "An introduction to variational methods for graphical models", In M.I. Jordan, ed., Learning in Graphical Models, Boston: Kluwer, 1998.

525. Jordan M.I. and R.A. Jacobs. "Modular and Hierarchical Learning Systems", in M.A. Arbib, ed., The Handbook of Brain Theory and Neural Networks, 1995, p. 579–583, Cambridge, MA: MIT Press.

526. Jordan M.I. and R.A. Jacobs. "Hierarchical mixtures of experts and the EM algorithm", Neural Computation, 1994, vol. 6, p. 181–214.

527. Jordan M.I. and R.A. Jacobs. "Hierarchies of adaptive experts", Advances in Neural Information Processing Systems, 1992, vol. 4, p. 985–992, San Mateo, CA: Morgan Kaufmann.

528. Joseph R.D. "The number of orthants in n-space intersected by an s-dimensional subspace", Technical Memo 8, Project PARA, Cornell Aeronautical Lab., Buffalo, NY, 1960.

529. Jutten C. and J. Heraulf. "Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture", Signal Processing, 1991, vol. 24, p. 1–10.

530. Kaas J.H., M.M. Merzenich and H.P. Killackey. "The reorganization of somatosensory cortex following peripheral nerve damage in adult and developing mammals", Annual Review of Neurosciences, 1983, vol. 6, p. 325–356.

531. Kailath T. Linear Systems, Englewood Cliffs, NJ: Prentice-Hall, 1980.

532. Kailath T. "A view of three decades of linear filtering theory", IEEE Transactions of Information Theory, 1974, vol. IT-20, p. 146–181.

533. Kailath T. "RKHS aproach to detection and estimation problems — Part I: Deterministic signals in Gaussian noise", IEEE Transactions of Information Theory, 1971, vol. IT-17, p. 530–549.

534. Kailath T. "An innovations aproach to least-squares estimation: Part 1. Linear filtering in additive white noise", IEEE Transactions of Automatic Control, 1968, vol. AC-13, p. 646–655.

535. Kalman R.E. "A new aproach to linear filtering and prediction problems", Transactions of the ASME, Journal of Basic Engineering, 1960, vol. 82, p. 35–45.

536. Kandel E.R. and J.H. Schwartz. Principles of Neural Science, 3rd ed., New York: Elsevier, 1991.

537. Kangas J., T. Kohonen and J. Laaksonen. "Variants of self-organizing maps", IEEE Transactions on Neural Networks 1, 1990, p. 93–99.

538. Kanter I. and H. Sompolinsky. "Associative recall of memory without errors", Physical Review A, 1987, vol. 35, p. 380–392.

539. Kaplen J.L. and J.A. Yorke. "Chaotic behavior of multidimensional difference equations", in H.-O Peitgen and H.-O Walker, eds., Functional Differential Equations and Aproximations of Fixed Points, 1979, p. 204–227, Berlin: Springer.

540. Kapen H.J. and F.B. Rodriguez. "Efficient learning in Boltzmann machines using linear response theory", Neural Computation, 1998, vol. 10.

541. Karhunen K. "Uber lineare methoden in der Wahrscheinlichkeitsrechnung", Annales Academiae Scientiarum Fennicae, Series AI: Mathematica-Physica, 1947, vol. 37, p. 3–79, (Transl.: RAND Corp., Santa Monica, CA, Rep.T-131, Aug. 1960).

542. Karhunen J. and J. Joutsensalo. "Generalizations of principal component analysis, optimization problems and neural networks", Neural Networks, 1995, vol. 8, p. 549–562.

543. Karpinski M. and A. Macintyre. "Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neuronal networks", Journal of Computer and System Sciences, 1997, vol. 54, p. 169–176.

544. Katagiri S. and E. McDermott. "Discriminative training — Recent progress in speech recognition", in C.H. Chen, L.F. Pau and P.S.P. Wang, eds., Handbook of Pattern Recognition and Computer Vision, 2nd edition, Singapore: World Scientific Publishing, 1996.

545. Katz B. Nerve, Muscle and Synapse, New York: McGraw-Hill, 1966.

546. Kawamoto A.H. and J.A. Anderson. "A neural network model of multistable perception", Acta Psychologica, 1985, vol. 59, p. 35–65.

547. Kawato M., H. Hayakama and T. Inui. "A forward-inverse optics model of reciprocal connections between visual cortical areas", Network, 1993, vol. 4, p. 415–422.

548. Kay J. "Feature discovery under contextual supervision using mutual information", International Joint Conference on Neural Networks, 1992, vol. IV, p. 79–84, Baltimore.

549. Kearns M. "A bound on the error of cross validation using the aproximation and estimation rates, with consequences for the training-test split", Advances in Neural Information Processing Systems, 1996, vol. 8, p. 183–189, Cambridge, MA: MIT Press.

550. Kearns M. and L.G. Valiant. "Cryptographic limitations on learning Boolean formulae and finite automata", Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, 1989, p. 433–444, New York.

551. Kearns M.J. and U.V. Vazirani. An Introduction to Computational Learning Theory, Cambridge, MA: MIT Press, 1994.

552. Kechriotis G., E. Zervas and E.S. Manolakos. "Using recurrent neural networks for adaptive communication channel equalization", IEEE Transactions on Neural Networks, 1994, vol. 5, p. 267–278.

553. Keeler J.D. "Basins of attraction of neural network models", in Neural Networks for Computing, J.S. Denker, ed., 1986, p. 259–264, New York: American Institute of Physics.

554. Keller J.B. "Inverse problems", American Mathematical Monthly, 1976, vol. 83, p. 107–118.

555. Kelso S.R., A.H. Ganong and T.H. Brown. "Hebbian synapses in hipocampus", Proceedings of the National Academy of Sciences, USA, 1986, vol. 83, p. 5326–5330.

556. Kennel M.B., R. Brown and H.D.I. Abarbanel. "Determining minimum embedding dimension using a geometrical construction", Physical Review A, 1992, vol. 45, p. 3403–3411.

557. Kerlirzin P. and F. Vallet. "Robustness in multilayer perceptrons", Neural Computation, 1993, vol. 5, p. 473–482.

558. Kirkpatrick S. "Optimization by simulated annealing: Quantitative Studies", Journal of Statistical Physics, 1984, vol. 34, p. 975–986.

559. Kirkpatrick S. and D. Sherrington. "Infinite-ranged models of spin-glasses", Physical Review, Series B, 1978, vol. 17, p. 4384–4403.

560. Kirkpatrick S., C.D. Gelatt, Jr. and M.P. Vecchi. "Optimization by simulated annealing", Science, 1983, vol. 220, p. 671–680.

561. Kirsch A. An Introduction to the Mathematical Theory of Inverse Problems, New York: Springer-Verlag, 1996.

562. Kleene S.C. "Representation of events in nerve nets and finite automata", in C.E. Shannon and J. McCarthy, eds., Automata Studies, Princeton, NJ: Princeton University Press, 1956.

563. Kmenta J. Elements of Econometrics, New York: Macmillan, 1971.

564. Knudsen E.I., S. duLac and S.D. Esterly. "Computational maps in the brain", Annual Review of Neuroscience, 1987, vol. 10, p. 41–65.

565. Koch C. and I. Segev, eds. Methods in Neuronal Modeling: From Synapses to Networks, Cambridge, MA: MIT Press, 1989.

566. Koch C., T. Poggio and V. Torre. "Nonlinear interactions in a dendritic tree: Localization, timing and role in information processing", Proceedings of the National Academy of Sciences, USA, 1983, vol. 80, p. 2799–2802.

567. Koch C. and B. Mathur. "Neuromorphic vision chips", IEEE Spectrum, 1996, vol. 33, no. 5, p. 38–46.

568. Kohonen T. "Exploration of very large databases by self-organizing maps", 1997 International Conference on Neural Networks, 1997, vol. I, p. PL1-PL6, Houston.

569. Kohonen T. Self-Organizing Maps, 2nd edition, Berlin: Springer-Verlag.

570. Kohonen T., 1996. "Emergence of invariant-feature detectors in the adaptive-subspace self-organizing maps", Biological Cybernetics, 1997Б, vol. 75, p. 281–291.

571. Kohonen T. "Physiological interpretation of the self-organizing map algorithm", Neural Networks, 1993, vol. 6, p. 895–905.

572. Kohonen T. "Things you haven't heard about the self-organizing map", Proceedings of the IEEE International Conference on neural networks, 1993, p. 1147–1156, San Francisco.

573. Kohonen T. "The self-organizing map", Proceedings of the Institute of Electrical and Electronics Engineers, 1990, vol. 78, p. 1464–1480.

574. Kohonen T. "Improved versions of learning vector quantization", IEEE International Joint Conference on Neural Networks, 1990Б, vol. I, p. 545–550, San Diego, CA.

575. Kohonen T. "An introduction to neural computing", Neural Networks, 1988, vol. 1, p. 3–16.

576. Kohonen T. Self-Organization and Associative Memory, 3rd edition, New York: Springer-Verlag, 1988Б.

577. Kohonen T. "The 'neural' phonetic typewriter", Computer, 1988б, vol. 21, p. 11–22.

578. Kohonen T. "Learning vector quantization for pattern recognition", Technical Report TKK-F-A601, Helsinki University of Technology, Finland, 1986.

579. Kohonen T. "Self-organized formation of topologically correct feature maps", Biological Cybernetics, 1982, vol. 43, p. 59–69.

580. Kohonen T. "Correlation matrix memories", IEEE Transactions on Computers, 1972, vol. C-21, p. 353–359.

581. Kohonen T. and E. Oja. "Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks for neuron-like elements", Biological Cybernetics, 1976, vol. 21, p. 85–95.

582. Kohonen T., E. Oja, O. Simula, A. Visa and J. Kangas. "Engineering aplications of the self-organizing map", Proceedings of the IEEE, 1996, vol. 84, p. 1358–1384.

583. Kohonen T., E. Reuhkala, K. Mäkisara and L. Vainio. "Associative recall of images", Biological Cybernetics, 1976, vol. 22, p. 159–168.

584. Kohonen T., G. Barna and R. Chrisley. "Statistical pattern recognition with neural networks: Benchmarking studies", IEEE International Conference on Neural Networks, 1988, vol. I, p. 61–68, San Diego, CA.

585. Kohonen T., J. Kangas, J. Laaksonen and K. Torkkola. "LVQ-PAK: The learning vector quantization Program Package", Helsinki University of Technology, Finland, 1992.

586. Koiran P. and E.D. Sontag. "Neural networks with quadratic VC dimension", Advances in Neural Information Processing Systems, 1996, vol. 8, p. 197–203, Cambridge, MA: MIT Press.

587. Kolen J.F. and J.B. Pollack. "Backpropagation is sensitive to initial conditions", Complex Systems, 1990, vol. 4, p. 269–280.

588. Kollias S. and D. Anastassiou. "An adaptive least squares algorithm for the efficient training of artificial neural networks", IEEE Transactions on Circuits and Systems, 1989, vol. 36, p. 1092–1101.

589. Kollias S. and D. Anastassiou. "Adaptive training of multilayer neural networks using a least squares estimation technique", IEEE International Conference on Neural Networks, 1988, vol. 1, p. 383–390, San Diego.

590. Kolmogorov A.N. "Interpolation and extrapolation of stationary random sequences", 1942, Rand Corporation, Santa Monica, CA., April 1962.

591. Kosko B. Fuzzy Engineering, Uper Saddle River, NJ: Prentice-Hall, 1997.

592. Kosko B. Neural Networks and Fuzzy Systems, Englewood Cliffs, NJ: Prentice-Hall, 1992.

593. Kosko B. "Bidirectional associative memories", IEEE Transactions on Systems, Man and Cybernetics, 1988, vol. 18, p. 49–60.

594. Kotilainen P. "Simulations and implementations of neural networks for principal component analysis", Electronics Lab Report 1-93, Tampre University of Technology, Finland, 1993.

595. Kraaijveld M.A. and R.P.W. Duin. "Generalization capabilities of minimal kernel-based networks", International Joint Conference on Neural Networks, 1991, vol. I, p. 843–848, Seattle.

596. Kramer A.H. and A. Sangiovanni-Vincentelli. "Efficient parallel learning algorithms for neural networks", Advances in neural Information Processing Systems, 1989, vol. 1, p. 40–48, San Mateo, CA: Morgan Kaufmann.

597. Kremer S.C. "Comments on constructive learning of recurrent neural networks: Limitations of recurrent cascade correlation and a simple solution", IEEE Transactions on Neural Networks, 1996, vol. 7, p. 1047–1049.

598. Kremer S.C. "On the computational power of Elman-style recurrent networks", IEEE Transactions on Neural Networks, 1995, vol. 6, p. 1000–1004.

599. Kreyszig E. Advanced Engineering Mathematics, 6th ed., New York: Wiley, 1988.

600. Krishnamurthy A.K., S.C. Ahalt, D.E. Melton and P. Chen. "Neural networks for vector quantization of speech and images", IEEE Journal of Selected Areas in Communications, 1990, vol. 8, p. 1449–1457.

601. Krzyzak A., T. Linder and G. Lugosi. "Nonparametric estimation and classification using radial basis functions", IEEE Transactions on Neural Networks, 1996, vol. 7, p. 475–487.

602. Kuan C.-M. and K. Hornik. "Convergence of learning algorithms with constant learning rates", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 484–489.

603. Kuan C.-M., K. Hornik and H. White. "A convergence result for learning in recurrent neural networks", Neural Computation, 1994, vol. 6, p. 420–440.

604. Kuffler S.W., J.G. Nicholls and A.R. Martin. From Neuron to Brain: A Cellular Aproach to the Function of the Nervous System, 2nd edition, Sunderland, MA: Sinauer Associates, 1984.

605. Kullback S. Information Theory and Statistics, Gloucester, MA: Peter Smith, 1968.

606. Kung S.Y. and K.I. Diamantaras. "A neural network learning algorithm for adaptive principal component extraction (APEX)", IEEE International Conference on Acoustics, Speech and Signal Processing, 1990, vol. 2, p. 861–864, Albuquerque.

607. Kushner H.J. and D.S. Clark. Stochastic Aproximation Methods for Constrained and Unconstrained Systems, New York: Springer-Verlag, 1978.

608. Lacoume J.L., P.O. Amblard and P. Comon. Statistiques d'ordre Superieurpour le Traitement du Signal, Masson Publishers, 1997.

609. Lancoz C. Linear Differential Operators, London: Van Nostrand, 1964.

610. Landau Y.D. Adaptive Control: The Model Reference Aproach, New York: Marcel Dekker, 1979.

611. Landa L.D. and E.M. Lifshitz. Statistical Physics: Part 1, 3rd edition, London: Pergamon Press, 1980.

612. Lanford O.E. "Strange attractors and turbulence", in H.L. Swinney and J.P. Gollub, eds., Hydrodynamic Instabilities and the Transition to Turbulence, New York: Springer-Verlag, 1981.

613. Lang K.J. and G.E. Hinton. "The development of the time-delay neural network architecture for speech recognition", Technical Report CMU-CS-88-152, Carnegie-Mellon University, Pittsburgh, PA, 1988.

614. Lapedes A. and R. Farber. "Programming a massively parallel, computation universal system: Static Behavior", In Neural Networks for Computing, J.S. Denker, ed., 1986, p. 283–298, New York: American Institute of Physics.

615. Larson J. and G. Lynch. "Theta pattern stimulation and the induction of LTP: The sequence in which synapses are stimulated determines the degree to which they potentiate", Brain Research, 1989, vol. 489, p. 49–58.

616. LaSalle J. and S. Lefschetz. Stability by Liapunov's Direct Method with Aplications, New York: Academic Press, 1961.

617. LeCun Y. Efficient Learning and Second-order Methods, A Tutorial at NIPS 93, Denver, 1993.

618. LeCun Y. "Generalization and network design strategies", Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto, Canada, 1989.

619. LeCun Y. "Une procedure d'aprentissage pour reseau a seuil assymetrique", Cognitiva, 1985, vol. 85, p. 599–604.

620. LeCun Y. and Y. Bengio. "Convolutional networks for images, speech and time series", in M.A. Arbib, ed., The Handbook of Brain Theory and Neural Networks, Cambridge, MA: MIT Press, 1995.

621. LeCun Y., B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard and L.D. Jackel. "Handwritten digit recognition with a back-propagation network", Advances in Neural Information Processing, 1990a, vol. 2, p. 396–404, San Mateo, CA: Morgan Kaufmann.

622. LeCun Y., L. Bottou and Y. Bengio. "Reading checks with multilayer graph transformeer networks", IEEE International Conference on Acoustics, Speech and Signal Processing, 1997, p. 151–154, Munich, Germany.

623. LeCun Y., L. Bottou, Y. Bengio and P. Haffner. "Gradient-based learning aplied to document recognition", Proceedings of the IEEE, 1998, vol. 86.

624. LeCun Y., J.S. Denker and S.A. Solla. "Optimal brain damage", Advances in Neural Information Processing Systems, 1990, vol. 2, p. 598–605, San Mateo, CA: Morgan Kaufmann.

625. LeCun Y., I. Kanter and S.A. Solla. "Second order properties of error surfaces: Learning time and generalization", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 918–924, Cambridge, MA: MIT Press.

626. Lee D.D. and H.S. Seung. "Unsupervised learning by convex and conic coding", Advances in Neural Information Processing Systems, 1997, vol. 9, p. 515–521, Cambridge, MA: MIT Press.

627. Lee T. Independent Component Analysis: Theory and Aplications, Ph.D. Thesis, Technische Universität, Berlin, Germany, 1997.

628. Lee T.-C., A.M. Peterson and J.J-C. Tsai. "A multilayer feed-forward neural network with dynamically adjustable structures", IEEE International Conference on Systems, Man and Cybernetics, 1990, p. 367–369, Los Angeles.

629. Lee Y. and R.P. Lipmann. "Practical characteristics of neural networks and conventional pattern classifiers on artificial and speech problems", Advances in Neural Information Processing Systems, 1990, vol. 2, p. 168–177, San Mateo, CA: Morgan Kaufmann.

630. Lee Y., S. Oh and M. Kim. "The effect of initial weights on premature saturation in back-propagation learning", International Joint Conference on Neural Networks, 1991, vol. I, p. 765–770, Seattle.

631. Lee Y.C., G. Doolen, H.H. Chan, G.Z. Sen, T. Maxwell, H.Y Lee and C.L. Giles. "Machine learning using a higher order correlation network", Physica, 1986, D22, p. 276–289.

632. Lefebvre W.C. An Object Oriented Aproach for the Analysis of Neural Networks, Master's Thesis, University of Florida, Gainsville, Fl, 1991.

633. Leon-Garcia A. Probability and Random Processes for Electrical Engineering, 2nd edition, Reading, MA: Addison-Wesley, 1994.

634. Leontaritis I. and S. Billings. "Input-output parametric models for nonlinear systems: Part I: Deterministic nonlinear systems", International Journal of Control, 1985, vol. 41, p. 303–328.

635. Levin A.V. and K.S. Narendra. "Control of nonlinear dynamical systems using neural networkя — Part II: Observability, identification and control", IEEE Transactions on Neural Networks, 1996, vol. 7, p. 30–42.

636. Levin A.V. and K.S. Narendra. "Control of nonlinear dynamical systems using neural networkя — Controllability and stabilization", IEEE Transactions on Neural Networks, 1993, vol. 4, p. 192–206.

637. Levine M. Man and Machine Vision. New York: McGraw-Hill, 1985.

638. Lewis F.L. and V.L. Syrmas. Optimal Control, 2nd edition, New York: Wiley (Interscience), 1995.

639. Lewis F.L., A. Yesildirek and K. Liu. "Multilayer neural-net robot controller with guaranteed tracking performance", IEEE Transactions on Neural Networks, 1996, vol. 7, p. 1–12.

640. Lichtenberg A.J. and M.A. Lieberman. Regular and Chaotic Dynamics, 2nd edition, New York: Springer-Verlag, 1992.

641. Light W.A. "Some aspects of radial basis function aproximation", in Aproximation Theory, Spline Functions and Aplications, S.P. Singh, ed., NATO ASI, 1992, vol. 256, p. 163–190, Boston: Kluwer Academic Publishers.

642. Light W. "Ridge functions, sigmoidal functions and neural networks", in E.W. Cheney, C.K. Chui and L.L. Schumaker, eds., Aproximation Theory VII, 1992Б, p. 163–206, Boston: Academic Press.

643. Lin J.K., D.G. Grier and J.D. Cowan. "Faithful representation of separable distributions", Neural Computation, 1997, vol. 9, p. 1305–1320.

644. Lin S. "Computer solutions of the traveling salesman problem", Bell System Technical Journal, 1965, vol. 44, p. 2245–2269.

645. Lin T., B.G. Horne, P. Tino and C.L. Giles. "Learning long-term dependencies in NARX recurrent neural networks", IEEE Transactions on Neural Networks, 1996, vol. 7, p. 1329–1338.

646. Linde Y., A. Buzo and R.M. Gray. "An algorithm for vector quantizer design", IEEE Transactions on Communications, 1980, vol. COM-28, p. 84–95.

647. Linsker R. "Deriving receptive fields using an optimal encoding criterion", Advances in Neural Information Processing Systems, 1993, vol. 5, p. 953–960, San Mateo, CA: Morgan Kaufmann.

648. Linsker R. "Designing a sensory processing system: What can be learned from principal components analysis"? Proceedings of the International Joint Conference on Neural Networks, 1990, vol. 2, p. 291–297, Washington, DC.

649. Linsker R. "Self-organization in a perceptual system: How network models and information theory may shed light on neural organization", Chapter 10 in Connectionist Modeling and Brain Function: The Developing Interface, S.J. Hanson and C.R. Olson, eds., 1990Б, p. 351–392, Cambridge, MA: MIT Press.

650. Linsker R. "Perceptual neural organization: Some aproaches based on network models and information theory", Annual Review of Neuroscience, 1990б, vol. 13, p. 257–281.

651. Linsker R. "An aplication of the principle of maximum information preservation to linear systems", Advances in Neural Information Processing Systems, 1989, vol. 1, p. 186–194, San Mateo, CA: Morgan Kaufmann.

652. Linsker R. "How to generate ordered maps by maximizing the mutual information between input and output signals", Neural computation, 1989Б, vol. 1, p. 402–411.

653. Linsker R. "Self-organization in a perceptual network", Computer, 1988, vol. 21, p. 105–117.

654. Linsker R. "Towards an organizing principle for a layered perceptual network", in Neural Information Processing Systems, D.Z. Anderson, ed., 1988Б, p. 485–494, New York: American Institute of Physics.

655. Linsker R. "Towards an organizing principle for perception: Hebbian synapses and the principle of optimal neural encoding", IBM Research Report RC12820, IBM Research, Yorktown Heights, NY, 1987.

656. Linsker R. "From basic network principles to neural architecture" (series), Proceedings of the National Academy of Sciences, USA, 1986, vol. 83, p. 7508–7512, 8390–8394, 8779–8783.

657. Lipmann R.P. "An introduction to computing with neural nets", IEEE ASSP Magazine, 1987, vol. 4, p. 4–22.

658. Lipmann R.P. "Review of neural networks for speech recognition", Neural Computation, 1989, vol. 1, p. 1–38.

659. Lipmann R.P. "Pattern classification using neural networks", IEEE Communications Magazine, 1989Б, vol. 27, p. 47–64.

660. Little W.A. "The existence of persistent states in the brain", Mathematical Biosciences, 1974, vol. 19, p. 101–120.

661. Little W.A. and G.L. Shaw. "Analytic study of the memory storage capacity of a neural network", Mathematical Biosciences, 1978, vol. 39, p. 281–290.

662. Little W.A. and G.L. Shaw. "A statistical theory of short and long term memory", Behavioral Biology, 1975, vol. 14, p. 115–133.

663. Livesey M. "Clamping in Boltzmann machines", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 143–148.

664. Ljung L. System Identification: Theory for the User. Englewood Cliffs, NJ: Prentice-Hall, 1987.

665. Ljung L. "Analysis of recursive stochastic algorithms", IEEE Transactions on Automatic Control, 1977, vol. AC-22, p. 551–575.

666. Ljung L. and T. Glad. Modeling of Dynamic Systems, Englewood Cliffs, NJ: Prentice-Hall, 1994.

667. Lloyd S.P. "Least squares quantization in PCM", неопубликованное техническое руководство лаборатории Bell Laboratories technical note. Позднее опубликовано в журнале IEEE Transactions on Information Theory, 1957, vol. IT-28, p. 127–135, 1982.

668. Lo Z.-P., M. Fujita and B. Bavarian. "Analysis of neighborhood interaction in Kohonen neural networks", 6th International Parallel Processing Symposium Proceedings, 1991, p. 247–249, Los Alamitos, CA.

669. Lo Z.-P., Y. Yu and B. Bavarian. "Analysis of the convergence properties of topology preserving neural networks", IEEE Transactions on Neural Networks, 1993, vol. 4, p. 207–220.

670. Lockery S.R., Y. Fang and T.J. Sejnowski. "A dynamical neural network model of sensorimotor transformations in the leech", International Joint Conference on Neural Networks, 1990, vol. I, p. 183–188, San Diego, CA.

671. Loéve M. Probability Theory, 3rd edition, New York: Van Nostrand, 1963.

672. Lorentz G.G. "The 13th problem of Hilbert", Proceedings of Symposia in Pure Mathematics, 1976, vol. 28, p. 419–430.

673. Lorentz G.G. Aproximation of Functions, Orlando, FL: Holt, Rinehart & Winston, 1966.

674. Lorenz E.N. "Deterministic non-periodic flows", Journal of Atmospheric Sciences, 1963, vol. 20, p. 130–141.

675. Lowe D. "Adaptive radial basis function nonlinearities and the problem of generalisation", First IEE International Conference on Artificial Neural Networks, 1989, p. 171–175, London.

676. Lowe D. "What have neural networks to offer statistical pattern processing"? Proceedings of the SPIE Conference on Adaptive Signal Processing, 1991, p. 460–471, San Diego, CA.

677. Lowe D. "On the iterative inversion of RBF networks: A statistical interpretation", Second IEE International Conference on Artificial Neural Networks, 1991Б, p. 29–33, Bournemouth, England.

678. Lowe D. and A.R. Webb. "Time series prediction by adaptive networks: A dynamical systems perspective", IEE Proceedings (London), Part F, 1991, vol. 138, p. 17–24.

679. Lowe D. and A.R. Webb. "Optimized feature extraction and the Bayes decision in feed-forward classifier networks", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991Б, PAMI-13, p. 355–364.

680. Lowe D. and A.R. Webb. "Exploiting prior knowledge in network optimization: an illustration from medical prognosis", Network, 1990, vol. 1, p. 299–323.

681. Lowe D. and M.E. Tiping. "Neuroscale: Novel topographic feature extraction using RBF networks", Neural Information Processing Systems, 1996, vol. 9, p. 543–549, Cambridge, MA: MIT Press.

682. Luenberger D.G. Linear and Nonlinear Programming, 2nd edition, Reading, MA: Addison-Wesley, 1984.

683. Lui H.C. "Analysis of decision contour of neural network with sigmoidal nonlinearity", International Joint Conference on Neural Networks, 1990, vol. I, p. 655–659, Washington, DC.

684. Luo Z. "On the convergence of the LMS algorithm with adaptive learning rate for linear feedforward networks", Neural Computation, 1991, vol. 3, p. 226–245.

685. Luo F. and R. Unbehauen. Aplied Neural Networks for Signal Processing, New York: Cambridge University Press, 1997.

686. Luttrell S.P. "A unified theory of density models and auto-encoders", Technical Report 97303, Defence Research Agency, Great Malvern, UK, 1997.

687. Luttrell S.P. "A Bayesian analysis of self-organizing maps", Neural Computation, 1994, vol. 6, p. 767–794.

688. Luttrell S.P. "Code vector density in topographic mapings: Scalar case", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 427–436.

689. Luttrell S.P. "Self-supervised training of hierarchical vector quantizers", 2nd International Conference on Artificial Neural Networks, 1991Б, p. 5–9, Bournemouth, England.

690. Luttrell S.P. "Hierarchical vector quantization", IEE Proceedings (London), 1989, vol. 136 (Part I), p. 405–413.

691. Luttrell S.P. "Self-organization: A derivation from first principle of a class of learning algorithms", IEEE Conference on Neural Networks, 1989Б, p. 495–498, Washington, DC.

692. Maass W. "Bounds for the computational power and learning complexity of analog neural nets", Proceedings of the 25th Annual ACM Symposium on the Theory of Computing, 1993, p. 335–344, New York: ACM Press.

693. Maass W. "Vapnik-Chervonenkis dimension of neural networks", in M.A. Arbib, ed., The Handbook of Brain Theory and Neural Networks, Cambridge, MA: MIT Press, 1993.

694. Mach E. "Über die Wirkung der räumlichen Vertheilung des Lichtreizes auf die Netzhaut, I. Sitzungsberichte der Mathematisch-Naturwissenschaftslichen Klasse der Kaiserlichen Akademie der Wissenschaften", 1865, vol. 52, p. 303–322.

695. MacKay D. "Bayesian interpolation", Neural Computation, 1992, vol. 4, p. 415–447.

696. MacKay D. "A practical Bayesian framework for back-propagation networks", Neural Computation, 1992Б, vol. 4, p. 448–472.

697. MacKay D.J.C. and K.D. Miller. "Analysis of Linsker's simulations of Hebbian rules", Neural Computation, 1990, vol. 2, p. 173–187.

698. Macintyre A.J. and E.D. Sontag. "Fitness results for sigmoidal 'neuronal' networks", Proceedings of the 25th Annual ACM Symposium on the Theory of Computing, 1993, p. 325–334, New York: ACM Press.

699. MacQueen J. "Some methods for classification and analysis of multivariate observation", in Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, L.M. LeCun and J. Neyman, eds., 1967, vol. 1, p. 281–297, Berkeley: University of California Press.

700. Madhuranath H. and S. Haykin. "Improved Activation Functions for Blind Separation: Details of Algebraic Derivations", CRL Internal Report No. 358, Communications Research Laboratory, McMaster University, Hamilton, Ontario, 1998.

701. Mahowald M.A. and C. Mead. "Silicon retina", in Analog VLSI and Neural Systems (C. Mead), Chapter 15. Reading, MA: Addison-Wesley, 1989.

702. Mandelbrot B.B. The Fractal Geometry of Nature, San Francisco: Freeman, 1982.

703. Mañé R. "On the dimension of the compact invariant sets of certain non-linear maps", in D. Rand and L.S. Young, eds., Dynamical Systems and Turbulence, Lecture Notes in Mathematics, 1981, vol. 898, p. 230–242, Berlin: Springer-Verlag.

704. Marr D. Vision, New York: W.H. Freeman and Company, 1982.

705. Martinetz T.M., H.J. Ritter and K.J. Schulten. "Three-dimensional neural net for learning visuomotor coordination of a robot arm", IEEE Transactions on Neural Networks, 1990, vol. 1, p. 131–136.

706. Mason S.J. "Feedback theory — Some properties of signal-flow graphs", Proceedings of the Institute of Radio Engineers, 1953, vol. 41, p. 1144–1156.

707. Mason S.J. "Feedback theory — Further properties of signal-flow graphs", Proceedings of the Institute of Radio Engineers, 1956, vol. 44, p. 920–926.

708. Maybeck P.S. Stochastic Models, Estimation and Control, vol. 2, New York: Academic Press, 1982.

709. Maybeck P.S. Stochastic Models, Estimation and Control, vol. 1, New York: Academic Press, 1979.

710. Mazaika P.K. "A mathematical model of the Boltzmann machine", IEEE First International Conference on Neural Networks, 1987, vol. III, p. 157–163, San Diego, CA.

711. McBride L.E., Jr. and K.S. Narendra. "Optimization of time-varying systems", IEEE Transactions on Automatic Contro, 1965l, vol. AC-10, p. 289–294.

712. McCullagh P. and J.A. Nelder. Generalized Linear Models, 2nd edition, London: Chapman and Hall, 1989.

713. McCulloch W.S. Embodiments of Mind, Cambridge, MA: MIT Press, 1988.

714. McCulloch W.S. and W. Pitts. "A logical calculus of the ideas immanent in nervous activity", Bulletin of Mathematical Biophysics, 1943, vol. 5, p. 115–133.

715. McEliece R.J., B.C. Posner, E.R. Rodemich and S.S. Venkatesh. "The capacity of the Hopfield associative memory", IEEE Transactions on Information Theory, 1987, vol. IT-33, p. 461–482.

716. McLachlan G.J. and K.E. Basford. Mixture Models: Inference and Aplications to Clustering, New York: Marcel Dekker, 1988.

717. McLachlan G.J. and T. Krishnan. The EM Algorithm and Extensions, New York: Wiley (Interscience), 1997.

718. McQueen J. "Some methods for classification and analysis of multivariate observations", Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967, vol. 1, p. 281–297, Berkeley, CA: University of California Press.

719. Mead C.A. "Neuromorphic electronic systems", Proceedings of the Institute of Electrical and Electronics Engineers, 1990, vol. 78, p. 1629–1636.

720. Mead C.A. Analog VLSI and Neural Systems, Reading, MA: Addison-Wesley, 1989.

721. Mead C.A. and M.A. Mahowald. "A silicon model of early visual processing", Neural Networks, 1988, vol. 1, p. 91–97.

722. Mead C.A., X. Arreguit and J. Lazzaro. "Analog VLSI model of binaural hearing", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 232–236.

723. Mecklenbräuker W. and F. Hlawatsch, eds. The Wigner Distribution, New York: Elsevier, 1997.

724. Memmi D. "Connectionism and artificial intelligence", Neuro-Nimes'89 International Workshop on Neural Networks and their Aplications, 1989, p. 17–34, Nimes, France.

725. Mendel J.M. Lessons in Estimation Theory for Signal Processing, Communications and Control. Englewood Cliffs, NJ: Prentice-Hall, 1995.

726. Mendel J.M. and R.W. McLaren. "Reinforcement-learning control and pattern recognition systems", in Adaptive, Learning and Pattern Recognition Systems: Theory and Aplications, vol. 66, J.M. Mendel and K.S. Fu, eds., 1970, p. 287–318, New York: Academic Press.

727. Mennon A., K. Mehrotra, C.K. Mohan and S. Ranka. "Characterization of a class of sigmoid functions with aplications to neural networks", Neural Networks, 1996, vol. 9, p. 819–835.

728. Mercer J. "Functions of positive and negative type and their connection with the theory of integral equations", Transactions of the London Philosophical Society (A), 1909, vol. 209, p. 415–446.

729. Mesulam M.M. "Attention, confusional states and neglect", in Principles of Behavioral Neurology, M.M. Mesulam, ed., Philadelphia: F.A. Davis, 1985.

730. Metropolis N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. Equations of state calculations by fast computing machines, Journal of Chemical Physics, 1953, vol. 21, p. 1087–1092.

731. Mhaskar H.N. "Neural networks for optimal aproximation of smooth and analytic functions", Neural Computation, 1996, vol. 8, p. 1731–1742.

732. Mhaskar H.N. and C.A. Micchelli. "Aproximation by superposition of sigmoidal and radial basis functions", Advances in Aplied Mathematics, 1992, vol. 13, p. 350–373.

733. Micchelli C.A. "Interpolation of scattered data: Distance matrices and conditionally positive definite functions", Constructive Aproximation, 1986, vol. 2, p. 11–22.

734. Miller D., A.V. Rao, K. Rose and A. Gersho. "A global optimization technique for statistical classifier design", IEEE Transactions on Signal Processing, 1996, vol. 44, p. 3108–3122.

735. Miller K.D., J.B. Keller and M.P. Stryker. "Ocular dominance column development: Analysis and simulation", Science, 1989, vol. 245, p. 605–615.

736. Miller D. and K. Rose. "Hierarchical, unsupervised learning with growing via phase transitions", Neural Computation, 1996, vol. 8, p. 425–450.

737. Miller D. and K. Rose. "Combined source-channel vector quantization using deterministic annealing", IEEE Transactions on Communications, 1994, vol. 42, p. 347–356.

738. Miller R. "Representation of brief temporal patterns, Hebbian synapses and the left-hemisphere dominance for phoneme recognition", Psychobiohgy, 1987, vol. 15, p. 241–247.

739. Minai A.A. and R.J. Williams. "Back-propagation heuristics: A study of the extended delta-bar-delta algorithm", IEEE International Joint Conference on Neural Networks, 1990, vol. I, p. 595–600, San Diego, CA.

740. Minsky M.L. Society of Mind, New York: Simon and Schuster, 1986.

741. Minsky M.L. Computation: Finite and Infinite Machines. Englewood Cliffs, NJ: Prentice-Hall, 1967.

742. Minsky M.L. "Steps towards artificial intelligence", Proceedings of the Institute of Radio Engineers, 1961, vol. 49, p. 8–30 (Reprinted in: Feigenbaum, E.A. and J. Feldman, eds., Computers and Thought, p. 406–450, New York: McGraw-Hill.)

743. Minsky M.L. "Theory of neural-analog reinforcement systems and its aplication to the brain-model problem", Ph.D. thesis, Princeton University, Princeton, NJ, 1954.

744. Minsky M.L. and S.A. Papert. Perceptrons, expanded edition, Cambridge, MA: MIT Press, 1988.

745. Minsky M.L. and S.A. Papert. Perceptrons, Cambridge, MA: MIT Press, 1969.

746. Minsky M.L. and O.G. Selfridge. "Learning in random nets", Information Theory, Fourth London Symposium, London: Butterworths, 1961.

747. Mitchell T.M. Machine Learning. New York: McGraw-Hill, 1997.

748. Mitchison G. "Learning algorithms and networks of neurons", in The Computing Neuron (R. Durbin, C. Miall and G. Michison, eds), 1989, p. 35–53, Reading, MA: Addison-Wesley.

749. Müller M.F. "A scaled conjugate gradient algorithm for fast supervised learning", Neural Networks, 1993, vol. 6, p. 525–534.

750. Moody J. and C.I. Darken. "Fast learning in networks of locally-tuned processing units", Neural Computation, 1989, vol. 1, p. 281–294.

751. Moody J. and L. Wu. "Optimization of trading systems and portfolios", in A. Weigend, Y. Abu-Mostafa and A.-P.N. Refenes, eds., Decision Technologies for Financial Engineering, 1996, p. 23–35, Singapore: World Scientific.

752. Moody J.E. and T. Rögnvaldsson. "Smoothing regularizers for projective basis function networks", Advances in Neural Information Processing Systems, 1997, vol. 9, p. 585–591.

753. Moray N. "Attention in dichotic listening: Affective cues and the influence of instructions", Quarterly Journal of Experimental Psychology, 1959, vol. 27, p. 56–60.

754. Morgan N. and H. Bourlard. "Continuous speech recognition using multilayer perceptrons with hidden Markov models", IEEE International Conference on Acoustics, Speech and Signal Processing, 1990, vol. 1, p. 413–416, Albuquerque.

755. Morita M. "Associative memory with nonmonotonic dynamics", Neural Networks, 1993, vol. 6, p. 115–126.

756. Morozov V.A. Regularization Methods for Ill-Posed Problems, Boca Raton, FL: CRC Press, 1993.

757. Morse P.M. and H. Feshbach. Methods of Theoretical Physics, Part 1, New York: McGraw-Hill, 1953.

758. Mozer M.C. "Neural net architectures for temporal sequence processing", in A.S. Weigend and N.A. Gershenfeld, eds., Time Series Prediction: Forecasting the Future and Understanding the Past, 1994, p. 243–264, Reading, MA: Addison-Wesley.

759. Mpitsos G.J. "Chaos in brain function and the problem of nonstationarity: A commentary", in Chaos in Brain Function, E. Basar, ed., 1990, p. 162–176. New York: Springer-Verlag.

760. M(ller B. and J. Reinhardt. Neural Networks: An Introduction, New York: Springer-Verlag, 1990.

761. Muller D. and G. Lynch. "Long-term potentiation differentially affects two components of synaptic responses in hipocampus", Proceedings of the National Academy of Sciences, USA, 1988, vol. 85, p. 9346–9350.

762. Mumford D. "Neural architectures for pattern-theoretic problems", in C. Koch and J. Davis, eds., Large-Scale Theories of the Cortex, 1994, p. 125–152, Cambridge, MA: MIT Press.

763. Murray M.K. and J.W. Rice. Differential Geometry and Statistics, New York: Chapman and Hall, 1993.

764. Murtagh B. and M. Saunders. "Large-scale linearly constrained optimization", Mathematical Programming, 1978, vol. 14, p. 41–72.

765. Muselli M. "On convergence properties of pocket algorithm", IEEE Transactions on Neural Networks, 1997, vol. 8, p. 623–629.

766. Nadal J.-P. and N. Parga. "Redundancy reduction and independent component analysis: Conditions on cumulants and adaptive aproaches", Neural Computation, 1997, vol. 9, p. 1421–1456.

767. Nadal J.-P. and N. Parga. "Nonlinear neurons in the low-noise limit: A factorial code maximizes information transfer", Network, 1994, vol. 5, p. 565–581.

768. Nadaraya É.A. "On nonparametric estimation of density functions and regression curves", Theory of Probability and its Aplications, 1965, vol. 10, p. 186–190.

769. Nadaraya È.A. "On estimating regression", Theory of Probability and its Aplications, 1964, vol. 9, p. 141–142.

770. Naftaly U., N. Intrator and D. Horn. "Optimal ensemble averaging of neural networks", Network, 1997, vol. 8, p. 283–296.

771. Nakano K. "Association — a model of associative memory", IEEE Transactions on Systems, Man and Cybernetics, 1972, vol. SMC-2, p. 380–388.

772. Narendra K.S. Neural Networks for Identification and Control, NIPS 95, Tutorial Program, 1995, p. 1–46, Denver.

773. Narendra K.S. and A.M. Annaswamy. Stable Adaptive Systems, Englewood Cliffs, NJ: Prentice-Hall, 1989.

774. Narendra K.S. and K. Parthasarathy. "Identification and control of dynamical systems using neural networks", IEEE Transactions on Neural Networks, 1990, vol. 1, p. 4–27.

775. Natarajan B.K. Machine Learning: A Theoretical Aproach, San Mateo, CA: Morgan Kaufmann, 1991.

776. Neal R.M. Bayesian Learning for Neural Networks, Ph.D. Thesis, University of Toronto, Canada, 1995.

777. Neal R.M. "Bayesian learning via stochastic dynamics", Advances in Neural Information Processing Systems, 1993, vol. 5, p. 475–482, San Mateo, CA: Morgan Kaufmann.

778. Neal R.M. "Connectionist learning of belief networks", Artificial Intelligence, 1992, vol. 56, p. 71–113.

779. Newcomb S. "A generalized theory of the combination of observations so as to obtain the best result", American Journal of Mathematics, 1886, vol. 8, p. 343–366.

780. Newell A. and H.A. Simon. Human Problem Solving, Englewood Cliffs, NJ: Prentice-Hall, 1972.

781. Ng K. and R.P. Lipmann. "Practical characteristics of neural network and conventional pattern classifiers", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 970–976, San Mateo, CA: Morgan Kaufmann.

782. Nguyen D. and B. Widrow. "The truck backer-uper: An example of self-learning in neural networks", International Joint Conference on Neural Networks, 1989, vol. II, p. 357–363, Washington, DC.

783. Nie J. and S. Haykin. "A Q-learning-based dynamic channel assignment technique for mobile communication systems", IEEE Transactions on Vehicular Technology, 1998

784. Nie J. and S. Haykin. "A dynamic channel assignment policy through Q-learning", CRL Report No. 334, Communications Research Laboratory, McMaster University, Hamilton, Ontario, 1996.

785. Nilsson N.J. Principles of Artificial Intelligence, New York: Springer-Verlag, 1980.

786. Nilsson N.J. Learning Machines: Foundations of Trainable Pattern-Classifying Systems, New York: McGraw-Hill, 1965.

787. Niyogi P. and F. Girosi. "On the relationship between generalization error, hypothesis complexity and sample complexity for radial basis functions", Neural Computation, 1996, vol. 8, p. 819–842.

788. Novikoff A.B.J. "On convergence proofs for perceptrons", Proceedings of the Symposium on the Mathematical Theory of Automata, 1962, p. 615–622, Brooklyn, NY: Polytechnic Institute of Brooklyn.

789. Nowlan S.J. "Maximum likelihood competitive learning", Advances in Neural Information Processing Systems, 1990, vol. 2, p. 574–582, San Mateo, CA: Morgan Kaufmann.

790. Nowlan S.J. and G.E. Hinton. "Adaptive soft weight tying using Gaussian mixtures", Advances in Neural Information Processing Systems, 1992, vol. 4, p. 993–1000, San Mateo, CA: Morgan Kaufmann.

791. Nowlan S.J. and G.E. Hinton. "Evaluation of adaptive mixtures of competing experts", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 774–780, San Mateo, CA: Morgan Kaufmann.

792. Obermayer K., H. Ritter and K. Schulten. "Development and spatial structure of cortical feature maps: A model study", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 11–17, San Mateo, CA: Morgan Kaufmann.

793. Oja E. "Principal components, minor components and linear neural networks", Neural Networks, 1992, vol. 5, 927–936.

794. Oja E. "Self-organizing maps and computer vision", Neural Networks for Perception, 1992Б, vol. 1, H. Wechsler, ed., vol. 1, p. 368–385, San Diego, CA: Academic Press.

795. Oja E. "Data compression, feature extraction and autoassociation in feedforward neural networks", Artificial Neural Networks, 1991, vol. 1, p. 737–746, Amsterdam: North-Holland.

796. Oja E. "Neural networks, principal components and subspaces", International Journal of Neural Systems, 1989, vol. 1, 61–68.

797. Oja E. Subspace Methods of Pattern Recognition, Letchworth, England: Research Studies Press, 1983.

798. Oja E. "A simplified neuron model as a principal component analyzer", Journal of Mathematical Biology, 1982, vol. 15, p. 267–273.

799. Oja E. and J. Karhunen. "A stochastic aproximation of the eigenvectors and eigenvalues of the expectation of a random matrix", Journal of Mathematical Analysis and Aplications, 1985, vol. 106, p. 69–84.

800. Oja E. and T. Kohonen. "The subspace learning algorithm as formalism for pattern recognition and neural networks", IEEE International Conference on Neural Networks, 1988, vol. I, p. 277–284, San Diego, CA.

801. Omlin C.W. and C.L. Giles. "Constructing deterministic finite-state automata in recurrent neural networks", Journal of the Association for Computing Machinery, 1996, vol. 43, p. 937–972.

802. Openheim A.V. and R.W. Schafer. Discrete-Time Signal Processing, Englewood Cliffs, NJ: PrenticeHall, 1989.

803. Orlando J., R. Mann and S. Haykin. "Classification of sea-ice using a dual-polarized radar", IEEE Journal of Oceanic Engineering, 1990, vol. 15, p. 228–237.

804. Osherson D.N., S. Weinstein and M. Stoli. "Modular learning", Computational Neuroscience, E.L. Schwartz, ed., 1990, p. 369–377, Cambridge, MA: MIT Press.

805. Osuna E. "Suport Vector Machines: Training and Aplications", Ph.D. Thesis, Operations Research Center, MIT, 1998.

806. Osuna E. and F. Girosi. "Reducing the run-time complexity of suport vector machines", ICPR 98, Brisbane, Australia, 1998.

807. Osuna E., R. Freund and F. Girosi. "An improved training algorithm for suport vector machines", Neural Networks for Signal Processing VII, Proceedings of the 1997 IEEE Workshop, 1997, p. 276–285, Amelia Island, FL.

808. Ott E. Chaos in Dynamical Systems, Cambridge, MA: Cambridge University Press, 1993.

809. Packard N.H., J.P. Crutchfield, J.D. Farmer and R.S. Shaw. "Geometry from a time series", Physical Review Letters, 1980, vol. 45, p. 712–716.

810. Palm G. Neural Assemblies: An Alternative Aproach, New York: Springer-Verlag, 1982.

811. Palmieri F. and S.A. Shah. "Fast training of multilayer perceptrons using multilinear parameterization", International Joint Conference on Neural Networks, 1990, vol. I, p. 696–699, Washington, DC.

812. Palmieri F., J. Zhu and C. Chang. "Anti-Hebbian learning in topologically constrained linear networks: A tutorial", IEEE Transactions on Neural Networks, 1993, vol. 5, p. 748–761.

813. Papoulis A. Probability, Random Variables and Stochastic Processes, 2nd edition, New York: McGraw-Hill, 1984.

814. Parisi G. Statistical Field Theory, Reading, MA: Addison-Wesley, 1988.

815. Park J. and I.W. Sandberg. "Universal aproximation using radial-basis-function networks", Neural Computation, 1991, vol. 3, p. 246–257.

816. Parker D.B. "Optimal algorithms for adaptive networks: Second order back propagation, second order direct propagation and second order Hebbian learning", IEEE 1st International Conference on Neural Networks, 1987, vol. 2, p. 593–600, San Diego, CA.

817. Parker D.B. "Learning-logic: Casting the cortex of the human brain in silicon", Technical Report TR-47, Center for Computational Research in Economics and Management Science, Cambridge, MA: MIT Press, 1985.

818. Parker T.S. and L.O., Chua. Practical Numerical Algorithms for Chaotic Systems, New York: Springer, 1989.

819. Parzen E. "On estimation of a probability density function and mode", Annals of Mathematical Statistics, 1962, vol. 33, p. 1065–1076.

820. Passino K.N. "Toward bridging the perceived gap between conventional and intelligent control", in M.D. Gupta and N.K. Sinha, eds., Intelligent Control Systems, 1996, p. 3–27, New York: IEEE Press.

821. Pavlov I.P. Conditional Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex, G.V. Anrep, New York: Oxford University Press, 1927.

822. Pearl J. Probabilistic Reasoning in Intelligent Systems, San Mateo, CA: Morgan Kaufmann, 1988 (Revised 2nd printing, 1991).

823. Pearlmutter B.A. "Learning state-space trajectories in recurrent neural networks", Neural Computation, 1989, vol. 1, p. 263–269.

824. Pearson K. "On lines and planes of closest fit to systems of points in space", Philosophical Magazine, 1901, vol. 2, p. 559–572.

825. Peretto P. "Collective properties of neural networks: A statistical physics aproach", Biological Cybernetics, 1984, vol. 50, p. 51–62.

826. Peretto P. and J.-J Niez. "Stochastic dynamics of neural networks", IEEE Transactions on Systems, Man and Cybernetics, 1986, vol. SMC-16, p. 73–83.

827. Perrin D. "Finite automata", in J. van Leeuwen, ed., Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, 1990, Chapter 1, p. 3–57, Cambridge, MA: MIT Press.

828. Perrone M.P. "Improving regression estimation: Averaging methods for variance reduction with extensions, to general convex measure optimization", Ph.D. Thesis, Brown University, Rhode Island, 1993.

829. Personnaz L., I. Guyon and G. Dreyfus. "Information storage and retrieval in spin-glass like neuralnetworks", Journal of Physique, Letters, Orsay, France, 1985, vol. 46, L-359–L-365.

830. Peterson C. "Mean field theory neural networks for feature recognition, content addressable memory and optimization", Connection Science, 1991, vol. 3, p. 3–33.

831. Peterson C. and J.R. Anderson. "A mean field theory learning algorithm for neural networks", Complex Systems, 1987, vol. 1, p. 995–1019.

832. Peterson C. and E. Hartman. "Explorations of the mean field theory learning algorithm", Neural Networks, 1989, vol. 2, p. 475–494.

833. Peterson C. and B. Söderberg. "A new method of maping optimization problems onto neural networks", International Journal of Neural Systems, 1989, vol. 1, p. 3–22.

834. Pham D.T. and P. Garrat. "Blind separation of mixture of independent sources through a quasi-maximum likelihood aproach", IEEE Transactions on Signal Processing, 1997, vol. 45, p. 1712–1725.

835. Pham D.T., P. Garrat and C. Jutten. "Separation of a mixture of independent sources through a maximum likelihood aproach", Proceedings of EUSIPCO, 1992, p. 771–774.

836. Phillips D. "A technique for the numerical solution of certain integral equations of the first kind", Journal of Association for Computing Machinery, 1962, vol. 9, p. 84–97.

837. Pineda F.J. "Recurrent backpropagation and the dynamical aproach to adaptive neural computation", Neural Computation, 1989, vol. 1, p. 161–172.

838. Pineda F.J. "Generalization of backpropagation to recurrent and higher order neural networks", in Neural Information Processing Systems, D.Z. Anderson, ed., 1988, p. 602–611, New York: American Institute of Physics.

839. Pineda F.J. "Dynamics and architecture in neural computation", Journal of Complexity, 1988Б, vol. 4, p. 216–245.

840. Pineda F.J. "Generalization of back-propagation to recurrent neural networks", Physical Review Letters, 1987, vol. 59, p. 2229–2232.

841. Pitts W. and W.S. McCulloch, "How we know universals: The perception of auditory and visual forms", Bulletin of Mathematical Biophysics, 1947, vol. 9, p. 127–147.

842. Plumbley M.D. and F. Fallside. "Sensory adaptation: An information-theoretic viewpoint", International Joint Conference on Neural Networks, 1989, vol. 2, p. 598, Washington, DC.

843. Plumbley M.D. and F. Fallside. "An information-theoretic aproach to unsupervised connectionist models", in Proceedings of the 1988 Connectionist Models Summer School, D. Touretzky, G. Hinton and T. Sejnowski, eds., 1988, p. 239–245. San Mateo, CA: Morgan Kaufmann.

844. Poggio T. "A theory of how the brain might work", Cold Spring Harbor Symposium on Quantitative Biology, 1990, vol. 5, p. 899–910.

845. Poggio T. and D. Beymer. "Learning to see", IEEE Spectrum, 1996, vol. 33, no. 5, p. 60–69.

846. Poggio T. and S. Edelman. "A network that learns to recognize three-dimensional objects", Nature, 1990, vol. 343, p. 263–266.

847. Poggio T. and F. Girosi. "Networks for aproximation and learning", Proceedings of the IEEE, 1990, vol. 78, p. 1481–1497.

848. Poggio T. and F. Girosi. "Regularization algorithms for learning that are equivalent to multilayer networks", Science, 1990Б, vol. 247, p. 978–982.

849. Poggio T. and C. Koch. "Ill-posed problems in early vision: From computational theory to analogue networks", Proceedings of the Royal Society of London, 1985, Series B, vol. 226, p. 303–323.

850. Poggio T., V. Torre and C. Koch. "Computational vision and regularization theory", Nature, 1985, vol. 317, p. 314–319.

851. Polak E. and G. Ribi(re. "Note sur la convergence de methods de directions conjuguees", Revue Fran(aise Information Recherche Operationnelle, 1969, vol. 16, p. 35–43.

852. Pöpel G. and U. Krey. "Dynamical learning process for recognition of correlated patterns in symmetric spin glass models", Europhysics Letters, 1987, vol. 4, p. 979–985.

853. Powell M.J.D. "The theory of radial basis function aproximation in 1990", in W. Light, ed., Advances in Numerical Analysis Vol. II: Wavelets, Subdivision Algorithms and Radial Basis Functions, 1992, p. 105–210, Oxford: Oxford Science Publications.

854. Powell M.J.D. "Radial basis function aproximations to polynomials", Numerical Analysis 1987 Proceedings, 1988, p. 223–241, Dundee, UK.

855. Powell M.J.D. "Radial basis functions for multivariable interpolation: A review", IMA Conference on Algorithms for the Aproximation of Functions and Data, 1985, p. 143–167, RMCS, Shrivenham, England.

856. Powell M.J.D. "Restart procedures for the conjugate gradient method", Mathematical Programming, 1977, vol. 12, p. 241–254.

857. Preisendorfer R.W. Principal Component Analysis in Meteorology and Oceanography, New York: Elsevier, 1988.

858. Press W.H., B.P. Flannery, S.A. Teukolsky and W.T. Vetterling. Numerical Recipes in C: The Art of Scientific Computing, Cambridge: Cambridge University Press, 1988.

859. Proakis J.G. Digital Communications, 2nd edition, New York: McGraw-Hill, 1989.

860. Prokhorov D.V. and D.C. Wunsch, II. "Adaptive critic designs", IEEE Transactions on Neural Networks, 1997, vol. 8, p. 997–1007.

861. Puskorius G.V. and L.A. Feldkamp. "Neurocontrol of nonlinear dynamical systems with Kalman filter-trained recurrent networks", IEEE Transactions on Neural Networks, 1994, vol. 5, p. 279–297.

862. Puskorius G.V. and L.A. Feldkamp. "Model reference adaptive control with recurrent networks trained by the dynamic DEKF algorithm", International Joint Conference on Neural Networks, 1992, vol. II, p. 106–113, Baltimore.

863. Puskorius G.V., L.A. Feldkamp and L.I. Davis, Jr. "Dynamic neural network methods aplied to onvehicle idle speed control", Proceedings of the IEEE, 1996, vol. 84, p. 1407–1420.

864. Puskorius G.V. and L.A. Feldkamp. "Decoupled extended Kalman filter training of feedforward layered networks", International Joint Conference on Neural Networks, 1991, vol. 1, p. 771–777, Seattle.

865. Rabiner L.R. "A tutorial on hidden Markov models", Proceedings of the IEEE, 1989, vol. 73, p. 1349–1387.

866. Rabiner L.R. and B.H. Juang. "An introduction to hidden Markkov models", IEEE ASSP Magazine, 1986, vol. 3, p. 4–16.

867. Rall W. "Cable theory for dendritic neurons", in Methods in Neuronal Modeling, C. Koch and I. Segev, eds., 1989, p. 9–62, Cambridge, MA: MIT Press.

868. Rall W. "Some historical notes", in Computational Neuroscience, E.L. Schwartz, Ed., 1990, p. 3–8, Cambridge: MIT Press.

869. Ramón y Cajál, S., Histologie du Systems Nerveux de l'homme et des vertébrés, Paris: Maloine, 1911.

870. Rao A., D. Miller K. Rose and A. Gersho. "Mixture of experts regression modeling by deterministic annealing", IEEE Transactions on Signal Processing, 1997, vol. 45, p. 2811–2820.

871. Rao A., K. Rose and A. Gersho. "A deterministic annealing aproach to discriminative hidden Markov model design", Neural Networks for Signal Processing VII, Proceedings of the 1997 IEEE Workshop, 1997Б, p. 266–275, Amelia Island, FL.

872. Rao C.R. Linear Statistical Inference and Its Aplications, New York: Wiley, 1973.

873. Rashevsky N. Mathematical Biophysics, Chicago: University of Chicago Press, 1938.

874. Raviv Y. and N. Intrator. "Bootstraping with noise: An effective regularization technique", Connection Science, 1996, vol. 8, p. 355–372.

875. Reed R. "Pruning algorithms — A survey", IEEE Transactions on Neural Networks, 1993, vol. 4, p. 740–747.

876. Reeke G.N. Jr., L.H. Finkel and G.M. Edelman. "Selective recognition automata", in An Introduction to Neural and Electronic Networks, S.F. Zornetzer, J.L. Davis and C. Lau, eds., 1990, p. 203–226, New York: Academic Press.

877. Reif. Fundamentals of Statistical and Thermal Physics, New York: McGraw-Hill, 1965.

878. Renals S. "Radial basis function network for speech pattern classification", Electronics Letters, 1989, vol. 25, p. 437–439.

879. Rènyi A. "On measures of entropy and information", Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability, 1960, p. 547–561.

880. Rènyi A. Probability Theory, North-Holland, Amsterdam, 1970.

881. Richard M.D. and R.P. Lipmann. "Neural network classifiers estimate Bayesian a posteriori probabilities", Neural Computation, 1991, vol. 3, p. 461–483.

882. Riesz F. and B. Sz-Nagy. Functional Analysis, 2nd edition, New York: Frederick Ungar, 1955.

883. Ripley B.D. Pattern Recognition and Neural Networks, Cambridge: Cambridge University Press, 1996.

884. Rissanen J. "Modeling by shortest data description", Automatica, 1978, vol. 14, p. 465–471.

885. Rissanen J. Stochastic Complexity in Statistical Inquiry, Singapore: World Scientific, 1989.

886. Ritter H. "Asymptotic level density for a class of vector quantization processes", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 173–175.

887. Ritter H. "Self-organizing feature maps: Kohonen maps", in M.A. Arbib, ed., The Handbook of Brain Theory and Neural Networks, 1995, p. 846–851, Cambridge, MA: MIT Press.

888. Ritter H. and T. Kohonen. "Self-organizing semantic maps", Biological Cybernetics, 1989, vol. 61, p. 241–254.

889. Ritter H. and K. Schulten. "Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability and dimension selection", Biological Cybernetics, 1988, vol. 60, p. 59–71.

890. Ritter H., T.M. Martinetz and K.J. Schulten. "Topology-conserving maps for learning visuo-motor-coordination", Neural Networks, 1989, vol. 2, p. 159–168.

891. Ritter H., T. Martinetz and K. Schulten. Neural Computation and Self-Organizing Maps: An Introduction, Reading, MA: Addison-Wesley, 1992.

892. Robbins H. and S. Monro. "A stochastic aproximation method", Annals of Mathematical Statistics, 1951, vol. 22, p. 400–407.

893. Robinson D.A. "Signal processing by neural networks in the control of eye movements", Computational Neuroscience Symposium, 1992, p. 73–78, Indiana University-Purdue University at Indianapolis.

894. Rochester N., J.H. Holland, L.H. Haibt and W.L. Duda. "Tests on a cell assembly theory of the action of the brain, using a large digital computer", IRE Transactions on Information Theory, 1956, vol. IT-2, p. 80–93.

895. Rose K. "Deterministic annealing for clustering, compression, classification, regression and related optimization problems", Proceedings of the IEEE, 1998, vol. 86.

896. Rose K. Deterministic Annealing, Clustering and Optimization, Ph.D. Thesis, California Institute of Technology, Pasadena, CA, 1991.

897. Rose K., E. Gurewitz and G.C. Fox. "Vector quantization by deterministic annealing", IEEE Transactions on Information Theory, 1992, vol. 38, p. 1249–1257.

898. Rose K., E. Gurewitz and G.C. Fox. "Statistical mechanics and phase transitions in clustering", Physical Review Letters, 1990, vol. 65, p. 945–948.

899. Rosenblatt F. Principles of Neurodynamics, Washington, DC: Spartan Books, 1962.

900. Rosenblatt F. "Perception simulation experiments", Proceedings of the Institute of Radio Engineers, 1960, vol. 48, p. 301–309.

901. Rosenblatt F. "On the convergence of reinforcement procedures in simple perceptrons", Cornell Aeronautical Laboratory Report, VG-1196–G-4, Buffalo, NY, 1960Б.

902. Rosenblatt F. "The Perceptron: A probabilistic model for information storage and organization in the brain", Psychological Review, 1958, vol. 65, p. 386–408.

903. Rosenblatt M. "Density estimates and Markov sequences", in M. Puri, ed., Nonparametric Techniques in Statistical Inference, 1970, p. 199–213, London: Cambridge University Press.

904. Rosenblatt M. "Remarks on some nonparametric estimates of a density function", Annals of Mathematical Statistics., 1956, vol. 27, p. 832–837.

905. Ross S.M. Introduction to Stochastic Dynamic Programming, New York: Academic Press, 1983.

906. Roth Z. and Y. Baram. "Multi-dimensional density shaping by sigmoids", IEEE Transactions on Neural Networks, 1996, vol. 7, p. 1291–1298.

907. Roussas G., ed. Nonparametric Functional Estimation and Related Topics, The Netherlands: Kluwer, 1991.

908. Roy S. and J.J. Shynk. "Analysis of the momentum LMS algorithm", IEEE Transactions on Acoustics, Speech and Signal Processing, 1990, vol. ASSP-38, p. 2088–2098.

909. Rubner J. and K. Schulten. "Development of feature detectors by self-organization", Biological Cybernetics, 1990, vol. 62, p. 193–199.

910. Rubner J. and P. Tavan. "A self-organizing network for principal component analysis", Europhysics Letters, 1989, vol. 10, p. 693–698.

911. Rueckl J.G., K.R. Cave and S.M. Kosslyn. "Why are 'what' and 'where' processed by separate cortical visual systems? A computational investigation", J. Cognitive Neuroscience, 1989, vol. 1, p. 171–186.

912. Rumelhart D.E. and J.L. McClelland, eds. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, Cambridge, MA: MIT Press, 1986.

913. Rumelhart D.E. and D. Zipser. "Feature discovery by competitive learning", Cognitive Science, 1985, vol. 9, p. 75–112.

914. Rumelhart D.E., G.E. Hinton and R.J. Williams. "Learning representations of back-propagation errors", Nature (London), 1986, vol. 323, p. 533–536.

915. Rumelhart D.E., G.E. Hinton and R.J. Williams. "Learning internal representations by error propagation", in D.E. Rumelhart and J.L. McCleland, eds., vol 1, Chapter 8, Cambridge, MA: MIT Press, 1986Б.

916. Russell S.J. and P. Novig. Artificial Intelligence: A Modern Aproach, Uper Saddle River, NJ: Prentice-Hall, 1995.

917. Russo A.P. Neural Networks for Sonar Signal Processing, Tutorial No. 8, IEEE Conference on Neural Networks for Ocean Engineering, Washington, DC, 1991.

918. Ruyck D.W., S.K. Rogers, M. Kabrisky, M.E. Oxley and B.W. Suter. "The multilayer perception as an aproximation to a Bayes optimal discriminant function", IEEE Transactions of Neural Networks, 1990, vol. 1, p. 296–298.

919. Saarinen S., R.B. Bramley and G. Cybenko. "Neural networks, backpropagation and automatic differentiation", in Automatic Differentiation of Algorithms: Theory, Implementation and Aplication, A. Griewank and G.F. Corliss, eds., 1992, p. 31–42, Philadelphia: SIAM.

920. Saarinen S., R. Bramley and G. Cybenko. "The numerical solution of neural network training problems", CRSD Report No. 1089, Center for Supercomputing Research and Development, University of Illinois, Urbana, IL, 1991.

921. Säckinger E., B.E. Boser, J. Bromley, Y. LeCun and L.D. Jackel. "Aplication of the ANNA neural network chip to high-speed character recognition", IEEE Transactions on Neural Networks, 1992, vol. 3, p. 498–505.

922. Säckinger E., B.E. Boser and L.D. Jackel. "A neurocomputer board based on the ANNA neural network chip", Advances in Neural Information Processing Systems, 1992Б, vol. 4, p. 773–780, San Mateo, CA: Morgan Kaufmann.

923. Saerens M. and A. Soquet. "Neural controller based on back-propagation algorithm", IEEE Proceedings (London), Part F, 1991, vol. 138, p. 55–62.

924. Sage A.P., ed. Concise Encyclopedia of Information Processing in Systems and Organizations, New York: Pergamon, 1990.

925. Salomon R. and J.L. van Hemmen. "Accelerating backpropagation through dynamic self-adaptation", Neural Networks, 1996, vol. 9, p. 589–601.

926. Samuel A.L. "Some studies in machine learning using the game of checkers", IBM Journal of Research and Development, 1959, vol. 3, p. 211–229.

927. Sandberg I.W. "Structure theorems for nonlinear systems", Multidimensional Systems and Signal Processing, 1991, vol. 2, p. 267–286.

928. Sandberg I.W., L. Xu. "Uniform aproximation of multidimensional myopic maps", IEEE Transactions on Circuits and Systems, 1997, vol. 44, p. 477–485.

929. Sandberg I.W. and L. Xu. "Uniform aproximation and gamma networks", Neural Networks, 1997Б, vol. 10, p. 781–784.

930. Sanger T.D. "Analysis of the two-dimensional receptive fields learned by the Hebbian algorithm in response to random input", Biological Cybernetics, 1990, vol. 63, p. 221–228.

931. Sanger T.D. "An optimality principle for unsupervised learning", Advances in Neural Information Processing Systems, 1989, vol. 1, p. 11–19, San Mateo, CA: Morgan Kaufmann.

932. Sanger T.D. "Optimal unsupervised learning in a single-layer linear feedforward neural network", Neural Networks, 1989Б, vol. 12, p. 459–473.

933. Sanner R.M. and J.,-J.E. Slotine. "Gaussian networks for direct adaptive control", IEEE Transactions on Neural Networks, 1992, vol. 3, p. 837–863.

934. Sauer N. "On the densities of families of sets", Journal of Combinatorial Theory, Series A, 1972, vol. 13, p. 145–172.

935. Sauer T., J.A. Yorke and M. Casdagli. "Embedology", Journal of Statistical Physics, 1991, vol. 65, p. 579–617.

936. Saul L.K., T. Jakkolla and M.I. Jordan. "Mean field theory for sigmoid belief networks", Journal of Artificial Intelligence Research, 1996, vol. 4, p. 61–76.

937. Saul L.K. and M.I. Jordan. "Exploiting tractable substructures in intractable networks", Advances in Neural Information Processing Systems, 1996, vol. 8, p. 486–492, Cambridge, MA: MIT Press.

938. Saul L.K. and M.I. Jordan. "Boltzmann chains and hidden Markov models", Advances in Neural Information Processing Systems, 1995, vol. 7, p. 435–442.

939. Schapire R.E. "Using output codes to boost multiclass learning problems", Machine Learning: Proceedings of the Fourteenth International Conference, Nashville, TN, 1997.

940. Schapire R.E. "The strength of weak learnability", Machine Learning, 1990, vol. 5, p. 197–227.

941. Schapire R.E., Y. Freund and P. Bartlett. "Boosting the margin: A new explanation for the effectiveness of voting methods", Machine Learning: Proceedings of the Fourteenth International Conference, Nashville, TN, 1997.

942. Schiffman W.H. and H.W. Geffers. "Adaptive control of dynamic systems by back propagation networks", Neural Networks, 1993, vol. 6, p. 517–524.

943. Schneider C.R. and H.C. Card. "Analog hardware implementation issues in deterministic Boltzmann machines", IEEE Transactions on Circuits and Systems II, 1998, vol. 45.

944. Schneider C.R. and H.C. Card. "Analog CMOS deterministic Boltzmann circuits", IEEE Journal Solid-State Circuits, 1993, vol. 28, p. 907–914.

945. Schülkopf B. Suport Vector Learning, Munich, Germany: R. Oldenbourg Verlag, 1997.

946. Schülkopf B., P. Simard, V. Vapnik and A.J. Smola. "Improving the accuracy and speed of suport vector machines", Advances in Neural Information Processing Systems, 1997, vol. 9, p. 375–381.

947. Schülkopf B., A. Smola and K.-R. Müller. "Nonlinear component analysis as a kernel eigenvalue problem", Neural Computation, 1998, vol. 10.

948. Schülkopf B., K.-K Sung, C.J.C. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik. "Comparing suport vector machines with Gaussian kernels to radial basis function classifiers", IEEE Transactions on Signal Processing, 1997, vol. 45, p. 2758–2765.

949. Schraudolph N.N. and T.J. Sejnowski. "Tempering back propagation networks: Not all weights are created equal", Advances in Neural Information Processing Systems, 1996, vol. 8, p. 563–569, Cambridge, MA: MIT Press.

950. Schumaker L.L. Spline Functions: Basic Theory, New York: Wiley, 1981.

951. Schurmars D. "Alternative metrics for maximum margin classification", NIPS Workshop on Suport Vector Machines, Beckenbridge, CO, 1997.

952. Schuster H.G. Deterministic Chaos: An Introduction, Weinheim, Germany: VCH, 1988.

953. Scofield C.L. and L.N. Cooper. "Development and properties of neural networks", Contemporary Physics, 1985, vol. 26, p. 125–145.

954. Scott A.C. Neurophysics, New York: Wiley, 1977.

955. Segee B.E. and M.J. Carter. "Fault tolerance of pruned multilayer networks", International Joint Conference on Neural Networks, 1991, vol. II, p. 447–452, Seattle.

956. Sejnowski T.J. "Strong covariance with nonlinearly interacting neurons", Journal of Mathematical Biology, 1977, vol. 4, p. 303–321.

957. Sejnowski T.J. "Statistical constraints on synaptic plasticity", Journal of Theoretical Biology, 1977Б, vol. 69, p. 385–389.

958. Sejnowski T.J. "On global properties of neuronal interaction", Biological Cybernetics, 1976, vol. 22, p. 85–95.

959. Sejnowski T.J. and P.S. Churchland. "Brain and cognition", in Foundations of Cognitive Science, M.I. Posner, ed., 1989, p. 301–356, Cambridge, MA: MIT Press.

960. Sejnowski T.J., P.K. Kienker and G.E. Hinton. "Learning symmetry groups with hidden units: Beyond the perceptron", Physica, 1986, vol. 22D, p. 260–275.

961. Sejnowski T.J., C. Koch and P.S. Churchland. "Computational neuroscience", Science, 1988, vol. 241, p. 1299–1306.

962. Sejnowski T.J. and C.R. Rosenberg. "Parallel networks that learn to pronounce English text", Complex Systems, 1987, vol. 1, p. 145–168.

963. Sejnowski T.J., B.P. Yuhas, M.H. Goldstein, Jr. and R.E. Jenkins. "Combining visual and acoustic speech signals with a neural network improves intelligibility", Advances in Neural Information Processing Systems, 1990, vol. 2, p. 232–239, San Mateo, CA: Morgan Kaufmann.

964. Selfridge O.G., R.S. Sutton and C.W. Anderson. "Selected bibliography on connectionism", Evolution, Learning and Cognition, Y.C. Lee, Ed., 1988, p. 391–403, River Edge, NJ: World Scientific Publishing, Inc.

965. Seung H. "Annealed theories of learning", in J.-H Oh, C. Kwon and S. Cho, eds., Neural Networks: The Statistical Mechanics Perspective, Singapore: World Scientific, 1995.

966. Seung H.S., T.J. Richardson, J.C. Lagarias and J.J. Hopfield. "Saddle point and Hamiltonian structure in excitatory-inhibitory networks", Advances in Neural Information Processing Systems, 1998, vol. 10.

967. Shah S. and F. Palmieri. "MEKA — A fast, local algorithm for training feedforward neural networks", International Joint Conference on Neural Networks, 1990, vol. 3, p. 41–46, San Diego, CA.

968. Shamma S. "Spatial and temporal processing in central auditory networks", in Methods in Neural Modeling, C. Koch and I. Segev, Eds., Cambridge, MA: MIT Press, 1989.

969. Shanno D.F. "Conjugate gradient methods with inexact line searches", Mathematics of Operations Research, 1978, vol. 3, p. 244–256.

970. Shannon C.E. "A mathematical theory of communication", Bell System Technical Journal, 1948, vol. 27. p. 379–423, 623–656.

971. Shannon C.E. and W. Weaver. The Mathematical Theory of Communication, Urbana, IL.: The University of Illinois Press, 1949.

972. Shannon C.E. and J. McCarthy, eds. Automata Studies, Princeton, NJ: Princeton University Press, 1956.

973. Shepherd G.M. Neurobiology, 2nd edition, New York: Oxford University Press, 1988.

974. Shepherd G.M. "Microcircuits in the nervous system", Scientific American, 1978, vol. 238, p. 92–103.

975. Shepherd G.M., ed. The Synoptic Organization of the Brain, 3rd edition, New York: Oxford University Press, 1990 (Ранее было опубликовано как отчет с результатами исследований Службы национальной безопасности; февраль, 1942.)

976. Shepherd G.M. "The significance of real neuron architectures for neural network simulations", in Computational Neuroscience, E.L. Schwartz, ed., 1990Б, p. 82–96, Cambridge: MIT Press.

977. Shepherd G.M. and C. Koch. "Introduction to synaptic circuits", in The Synoptic Organization of the Brain, G.M. Shepherd, ed., 1990, p. 3–31. New York: Oxford University Press.

978. Sherrington C.S. The Integrative Action of the Nervous System, New York: Oxford University Press, 1906.

979. Sherrington C.S. The Brain and Its Mechanism, London: Cambridge University Press, 1933.

980. Sherrington D. and S. Kirkpatrick. "Spin-glasses", Physical Review Letters, 1975, vol. 35, p. 1972.

981. Shewchuk J.R. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, 1994, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 4, 1994.

982. Shore J.E. and R.W. Johnson. "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy", IEEE Transactions on Information Theory, 1980, vol. IT-26, p. 26–37.

983. Shynk J.J. "Performance surfaces of a single-layer perceptron", IEEE Transactions on Neural Networks, 1990, 1, p. 268–274.

984. Shynk J.J. and N.J. Bershad. "Steady-state analysis of a single-layer perceptron based on a system identification model with bia terms", IEEE Transactions on Circuits and Systems, 1991, vol. CAS-38, p. 1030–1042.

985. Shustorovich A. "A subspace projection aproach to feature extraction: The two-dimensional Gabor transform for character recognition", Neural Networks, 1994, vol. 7, p. 1295–1301.

986. Shustorovich A. and C. Thrasher. "Neural network positioning and classification of handwritten characters", Neural Networks, 1996, vol. 9, p. 685–693.

987. Shynk J.J. and N.J. Bershad. "Stationary points and performance surfaces of a perceptron learning algorithm for a nonstationary data model", International Joint Conference on Neural Networks, 1992, vol. 2, p. 133–139, Baltimore.

988. Shynk J.J. and N.J. Bershad. "Steady-state analysis of a single-layer perceptron based on a system identification model with bias terms", IEEE Transactions on Circuits and Systems, 1991, vol. CAS-38, p. 1030–1042.

989. Siegelmann H.T., B.G. Horne and C.L. Giles. "Computational capabilities of recurrent NARX neural networks", Systems, Man and Cybernetics, Part B: Cybernetics, 1997, vol. 27, p. 208–215.

990. Siegelmann H.T. and E.D. Sontag. "Turing computability with neural nets", Aplied Mathematics Letters, 1991, vol. 4, p. 77–80.

991. Simard P., Y. LeCun and J. Denker. "Efficient pattern recognition using a new transformation distance", Advances in Neural Information Processing Systems, 1993, vol. 5, p. 50–58, San Mateo, CA: Morgan Kaufmann.

992. Simard P., B. Victorri, Y. LeCun and J. Denker. "Tangent prop — A formalism for specifying selected invariances in an adaptive network", Advances in Neural Information Systems, 1992, vol. 4, p. 895–903, San Mateo, CA: Morgan Kaufmann.

993. Simmons J.A. "A view of the world through the bat's ear: The formation of acoustic images in echolocation", Cognition, 1989, vol. 33, p. 155–199.

994. Simmons J.A., P.A. Saillant and S.P. Dear. "Through a bat's ear", IEEE Spectrum, 1992, vol. 29(3), p. 46–48.

995. Singh S.P., ed. Aproximation Theory, Spline Functions and Aplications, Dordrecht, The Netherlands: Kluwer, 1992.

996. Singh S. and D. Bertsekas. "Reinforcement learning for dynamic channel allocation in cellular telephone systems", Advances in Neural Information Processing Systems, 1997, vol. 9, p. 974–980, Cambridge, MA: MIT Press.

997. Singhal S. and L. Wu. "Training feed-forward networks with the extended Kalman filter", IEEE International Conference on Acoustics, Speech and Signal Processing, 1989, p. 1187–1190, Glasgow, Scotland.

998. Singleton R.C. "A test for linear separability as aplied to self-organizing machines", in M.C. Yovitz, G.T. Jacobi and G.D. Goldstein, eds., Self Organizing Systems, 1962, p. 503–524, Washington DC: Spartan Books.

999. Sjöberg J., Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson and A. Juditsky. "Nonlinear black-box modeling in system identification: A unified overview", Automatica, 1995, vol. 31, p. 1691–1724.

1000. Slepian D. Key papers in the development of information theory, New York: IEEE Press, 1973.

1001. Sloane N.J.A. and A.D. Wyner. Claude Shannon: Collected Papers, New York: IEEE Press, 1993.

1002. Slotine J.-J. and W. Li. Aplied Nonlinear Control, Englewood Cliffs, NJ: Prentice-Hall, 1991.

1003. Smith M. Neural Networks for Statistical Modeling, New York: Van Nostrand Reinhold, 1993.

1004. Smola A.J. and B. Sch(lkopf. "From regularization operators to suport vector kernels", Advances in Neural Information Processing Systems, 1998, vol. 10.

1005. Smolensky P. "On the proper treatment of connectionism", Behavioral and Brain Sciences, 1988, vol. 11, p. 1–74.

1006. Sontag E.D. "Recurrent neural networks: Some learning and systems-theoretic aspects", Department of Mathematics, Rutgers University, New Brunswick, NJ, 1996.

1007. Sontag E.D. "Feedback stabilization using two-hidden-layer nets", IEEE Transactions on Neural Networks, 1992, vol. 3, p. 981–990.

1008. Sontag E.D. Mathematical Control Theory: Deterministic Finite Dimensional Systems, New York: Springer-Verlag, 1990.

1009. Sontag E.D. "Sigmoids distinguish more efficiently than Heavisides", Neural Computation, 1989, vol. 1, p. 470–472.

1010. Southwell R.V. Relaxation Methods in Theoretical Physics, New York: Oxford University Press, 1946.

1011. Specht D.F. "A general regression neural network", IEEE Transactions on Neural Networks, 1991, vol. 2, p. 568–576.

1012. Sperduti A. "On the computational power of recurrent neural networks for structures", Neural Networks, 1997, vol. 10, p. 395–400.

1013. Sperduti A. and A. Starita. "Supervised neural networks for the classification of structures", IEEE Transactions on Neural Networks, 1997, vol. 8, p. 714–735.

1014. Sprecher D.A. "On the structure of continuous functions of several variables", Transactions of the American Mathematical Society, 1965, vol. 115, p. 340–355.

1015. Steinbuch K. "Die Lernmatrix", Kybernetik, 1961, vol. 1, p. 36–45.

1016. Stent G.S. "A physiological mechanism for Hebb's postulate of learning", Proceedings of the National Academy of Sciences, USA, 1973, vol. 70, p. 997–1001.

1017. Sterling P. "Retina", in The Synaptic Organization of the Brain, G.M. Shepherd, ed., 3rd edition, p. 170–213, New York: Oxford University Press, 1990.

1018. Stevenson M., R. Winter and B. Widrow. "Sensitivity of layered neural networks to errors in the weights", International Joint Conference on Neural Networks, 1990, vol. 1, p. 337–340, Washington, DC.

1019. Stone M. "Cross-validation: A review", Mathematische Operationsforschung Statistischen, Serie Statistics, 1978, vol. 9, p. 127–139.

1020. Stone M. "Cross-validatory choice and assessment of statistical predictions", Journal of the Royal Statistical Society, 1974, vol. B36, p. 111–133.

1021. Stork D. "Is backpropagation biologically plausible"? International Joint Conference on Neural Networks, 1989, vol. 2, p. 241–246, Washington, DC.

1022. Strang G. Linear Algebra and its Aplications, New York: Academic Press, 1980.

1023. Stuart A. and K. Ord. Kendall's Advanced Theory of Statistics, vol. I, 6th edition, New York: Halsted Press, 1994.

1024. Su H.-T. and T. McAvoy. "Identification of chemical processes using recurrent networks", Proceedings of the 10th American Controls Conference, 1991, vol. 3, p. 2314–2319, Boston.

1025. Su H.-T., T. McAvoy and P. Werbos. "Long-term predictions of chemical processes using recurrent neural networks: A parallel training aproach", Industrial Engineering and Chemical Research, 1992, vol. 31, p. 1338–1352.

1026. Suga N. "Cortical computational maps for auditory imaging", Neural Networks, 1990, vol. 3, p. 3–21.

1027. Suga N. "Computations of velocity and range in the bat auditory system for echo location", in Computational Neuroscience, E.L. Schwartz, ed., p. 213–231, Cambridge, MA: MIT Press, 1990Б.

1028. Suga N. "Biosonar and neural computation in bats", Scientific American, 1990б, vol. 262, p. 60–68.

1029. Suga N. "The extent to which bisonar information is represented in the bat auditory cortex", in Dynamic Aspects of Neocortical Function, G.M. Edelman, WE. Gall and W.M. Cowan, eds. p. 653–695, New York: Wiley (Interscience), 1985.

1030. Suga N. and J.S. Kanwal. "Echolocation: Creating computational maps", in M.A. Arbib, ed., The Handbook of Brain Theory and Neural Networks, Cambridge, MA: MIT Press, 1995.

1031. Sutton J.P. and J.A. Anderson. "Computational and neurobiological features of a network of networks", in J.M. Bower, ed., The Neurobiology of Computation, p. 317–322, Boston: Kluwer, 1995.

1032. Sutton R.S. "Learning to predict by the methods of temporal differences", Machine Learning, 1988, vol. 3, p. 9–44.

1033. Sutton R.S. "Two problems with back-propagation and other steepest-descent learning procedures for networks", Proceedings of the Eighth Annual Conference of the Cognitive Science Society, p. 823–831. Hillsdale, NJ: Lawrence Erlbaum, 1986.

1034. Sutton R.S., ed. Special Issue on Reinforcement Learning, Machine Learning, 1992, vol. 8, p. 1–395.

1035. Sutton R.S. "Temporal credit assignment in reinforcement learning", Ph.D. Dissertation, University of Massachusetts, Amherst, MA, 1984.

1036. Sutton R.S. and A.G. Barto. Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press, 1998.

1037. Suykens J.A.K., J.P.L. Vandewalle and B.L.R. DeMoor. Artificial Neural Networks for Modeling and Control of Non-Linear Systems, Dordrecht, The Netherlands: Kluwer, 1996.

1038. Swindlehurst A.L., M.J. Goris and B. Ottersten. "Some experiments with array data collected in actual urban and suburban environments", IEEE Workshop on Signal Processing Advances in Wireless Communications, p. 301–304, Paris, France, 1997.

1039. Takahashi Y. "Generalization and aproximation capabilities of multilayer networks", Neural Computation, 1993, vol. 5, p. 132–139.

1040. Takens F. "On the numerical determination of the dimension of an attractor", in D. Rand and L.S. Young, eds., Dynamical Systems and Turbulence, Annual Notes in Mathematics, 1981, vol. 898, p. 366–381, Berlin: Springer-Verlag.

1041. Tapia R.A. and J.R. Thompson. Nonparametric Probability Density Estimation, Baltimore:The Johns Hopkins University Press, 1978.

1042. Taylor J.G. "Neural computation: The historical background", in E. Fiesler and R. Beale, eds., Handbook of Neural Computation, New York: Oxford University Press, 1997.

1043. Taylor W.K. "Cortico-thalamic organization and memory", Proceedings of the Royal Society, London, Series B, 1964, vol. 159, p. 466–478.

1044. Taylor W.K. "Electrical simulation of some nervous system functional activities", Information Theory, 1956, vol. 3, E.C. Cherry, ed., p. 314–328, London: Butterworths.

1045. Tesauro G. "Temporal difference learning and TD-gamma", Communications of the Association for Computing Machinery, 1995, vol. 38, p. 58–68.

1046. Tesauro G. "TD-Gammon, A self-teaching Backgammon program, achieves master-level play", Neural Computation, 1994, vol. 6, p. 215–219.

1047. Tesauro G. "Practical issues in temporal difference learning", Machine Learning, 1992, vol. 8, p. 257–277.

1048. Tesauro G. "Neurogammon wins computer olympiad", Neural Computation, 1989, vol. 1, p. 321–323.

1049. Tesauro G. and T.J. Sejnowski. "A parallel network that learns to play backgamma", Artificial Intelligence, 1989, vol. 39, p. 357–390.

1050. Tesauro G. and R. Janssens. "Scaling relationships in back-propagation learning", Complex Systems, 1988, vol. 2, p. 39–44.

1051. Teyler T.J. "Memory: Electrophysiological analogs", in Learning and Memory: A Biological View, J.L. Martinez, Jr. and R.S. Kesner, eds., p. 237–265, New York: Academic Press, 1986.

1052. Thorndike E.L. Animal Intelligence, Darien, CT: Hafner, 1911.

1053. Thrun S.B. "The role of exploration in learning control", in Handbook of Intelligent Control, D.A. White and D.A. Sofge, eds., p. 527–559, New York: Van Nostrand Reinhold, 1992.

1054. Tikhonov A.N. "On regularization of ill-posed problems", Doklady Akademii Nauk USSR, 1973, vol. 153, p. 49–52.

1055. Tikhonov A.N. "On solving incorrectly posed problems and method of regularization", Doklady Akademii Nauk USSR, 1963, vol. 151, p. 501–504.

1056. Tikhonov A.N. and V.Y. Arsenin. Solutions of Ill-posed Problems, Washington, DC: W.H. Winston, 1977.

1057. Titterington D.M., A.F.M. Smith and V.E. Makov. Statistical Analysis of Finite Mixture Distributions, New York: Wiley, 1985.

1058. Touretzky D.S. and D.A. Pomerleau. "What is hidden in the hidden layers"? Byte, 1989, vol. 14, p. 227–233.

1059. Tsitsiklis J.N. "Asynchronous stochastic aproximation and Q-learning", Machine Learning, 1994, vol. 16, p. 185–202.

1060. Tsoi A.C. and A.D. Back. "Locally recurrent globally feedforward networks: A critical review", IEEE Transactions on Neural Networks, 1994, vol. 5, p. 229–239.

1061. Turing A.M. "The chemical basis of morphogenesis", Philosophical Transactions of the Royal Society, B, 1952, vol. 237, p. 5–72.

1062. Turing A.M. "Computing machinery and intelligence", Mind, 1950, vol. 59, p. 433–460.

1063. Turing A.M. "On computable numbers with an aplication to the Entscheidungs problem", Proceedings of the London Mathematical Society, Series 2, 1936, vol. 42, p. 230–265; Исправления опубликованы в этом же издании, vol. 43, p. 544–546.

1064. Tsoi A.C. and A. Back. "Locally recurrent globally feedforward networks: A critical review", IEEE Transactions on Neural Networks, 1994, vol. 5, p. 229–239.

1065. Tzefestas S.G., ed. Methods and Aplications of Intelligent Control, Boston: Kluwer, 1997.

1066. Udin S.B. and J.W. Fawcett. "Formation of topographic maps", Annual Review of Neuroscience, 1988, vol. 2, p. 289–327.

1067. Ukrainec A.M. and S. Haykin. "A modular neural network for enhancement of cross-polar radar targets", Neural Networks, 1996, vol. 9, p. 143–168.

1068. Ukrainec A. and S. Haykin. "Enhancement of radar images using mutual information based unsupervised neural networks", Canadian Conference on Electrical and Computer Engineering, p. MA6.9.1–MA6.9.4, Toronto, Canada, 1992.

1069. Uttley A.M. Information Transmission in the Nervous System, London: Academic Press, 1979.

1070. Uttley A.M. "The informon: A network for adaptive pattern recognition", Journal of Theoretical Biology, 1970, vol. 27, p. 31–67.

1071. Uttley A.M. "The transmission of information and the effect of local feedback in theoretical and neural networks", Brain Research, 1966, vol. 102, p. 23–35.

1072. Uttley A.M. "A theory of the mechanism of learning based on the computation of conditional probabilities", Proceedings of the First International Conference on Cybernetics, Namur, Gauthier-Villars, Paris, 1956.

1073. Vaillant R., C. Monrocq and Y. LeCun. "Original aproach for the localization of objects in images", IEEE Proceedings (London) on Vision, Image and Signal Processing, 1994, vol. 141, p. 245–250.

1074. Valavanis K.P. and G.N. Saridis. Intelligent Robotic Systems: Theory, Design and Aplications, Norwell. MA: Kluwer, 1992.

1075. Valiant L.G. "A theory of the learnable", Communications of the Association for Computing Machinery, 1984, vol. 27, p. 1134–1142.

1076. Vanderbei R. "Interior point methods: Algorithms and formulations", ORSA Journal on Computing, 1994, vol. 6, p. 32–34.

1077. Van Essen D.C., C.H. Anderson and D.J. Felleman. "Information processing in the primate visual system: An integrated systems perspective", Science, 1992, vol. 255, p. 419–423.

1078. Van de Laar P., T. Heskes and S. Gielen. "Task-dependent learning of attention", Neural Networks, 1997, vol. 10, p. 981–992.

1079. Van Laarhoven P.J.M. and E.H.L. Aarts. Simulated Annealing: Theory and Aplications, Boston: Kluwer Academic Publishers, 1988.

1080. Van Trees H.L. Detection, Estimation and Modulation Theory, Part I, New York: Wiley, 1968.

1081. Van Hulle M.M. "Nonparametric density estimation and regression achieved with topographic maps maximizing the information-theoretic entropy of their outputs", Biological Cybernetics, 1997, vol. 77, p. 49–61.

1082. Van Hulle M.M. "Topographic map formation by maximizing unconditional entropy: A plausible strategy for "on-line" unsupervised competitive learning and nonparametric density estimation", IEEE Transactions on Neural Networks, 1996, vol. 7, p. 1299–1305.

1083. Van Veen B. "Minimum variance beamforming", in S. Haykin and A. Steinhardt, eds., Adaptive Radar Detection and Estimation, New York: Wiley (Interscience), 1992.

1084. Vapnik V.N. Statistical Learning Theory, New York: Wiley, 1998.

1085. Vapnik V.N. The Nature of Statistical Learning Theory, New York: Springer-Verlag, 1995.

1086. Vapnik V.N. "Principles of risk minimization for learning theory", Advances in Neural Information Processing Systems, 1992, vol. 4, p. 831–838, San Mateo, CA: Morgan Kaufmann.

1087. Vapnik V.N. Estimation of Dependences Based on Empirical Data, New York: Springer-Verlag, 1982.

1088. Vapnik V.N. and A.Ya. Chervonenkis. "On the uniform convergence of relative frequencies of events to their probabilities", Theoretical Probability and Its Aplications, 1971, vol. 17, p. 264–280.

1089. Vapnik V.N. and A.Ya. Chervonenkis. "A note on a class of perceptrons", Automation and Remote Control, 1964, vol. 25, p. 103–109.

1090. Velmans M. "Consciousness, Theories of", In M.A. Arbib, ed., The Handbook of Brain Theory and Neural Networks, p. 247–250, Cambridge, MA: MIT Press, 1995.

1091. Venkataraman S. "On encoding nonlinear oscillations in neural networks for locomotion", Proceedings of the 8th Yale Workshop on Adaptive and Learning Systems, p. 14–20, New Haven, CT, 1994.

1092. Venkatesh S.J., G. Panche, D. Psaltis and G. Sirat. "Shaping attraction basins in neural networks", Neural Networks, 1990, vol. 3, p. 613–623.

1093. Vetterli M. and J. Kovačevic'. Wavelets and Subband Coding, Englewood Cliffs, NJ: Prentice-Hall, 1995.

1094. Vidyasagar M. A Theory of Learning and Generalization, London: Springer-Verlag, 1997.

1095. Vidyasagar M. Nonlinear Systems Analysis, 2nd edition, Englewood Cliffs, NJ: Prentice-Hall, 1993.

1096. Viterbi A.J. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", IEEE Transactions on Information Theory, 1967, vol. IT-13, p. 260–269.

1097. von der Malsburg C. "Network self-organization", in An Introduction to Neural and Electronic Networks, S.F. Zornetzer, J.L. Davis and C. Lau, eds., p. 421–432, San Diego, CA: Academic Press, 1990.

1098. von der Malsburg C. "Considerations for a visual architecture", in Advanced Neural Computers, R. EckMiller ed., p. 303–312, Amsterdam: North-Holland, 1990Б.

1099. von der Malsburg C. "The correlation theory of brain function", Internal Report 81–2, Department of Neurobiology, Max-Plak-Institute for Biophysical Chemistry, G(ttingen, Germany, 1981.

1100. von der Malsburg C. "Self-organization of orientation sensitive cells in the striate cortex", Kybernetik, 1973, vol. 14, p. 85–100.

1101. von der Malsburg C. and W. Schneider. "A neural cocktail party processor", Biological Cybernetics, 1986, vol. 54, p. 29–40.

1102. von Neumann J. Papers of John von Neumann on Computing and Computer Theory, W. Aspray and A. Burks, eds., Cambridge, MA: MIT Press, 1986.

1103. von Neumann J. The Computer and the Brain, New Haven, CT: Yale University Press, 1958.

1104. von Neumann J. "Probabilistic logics and the synthesis of reliable organisms from unreliable components", in Automata Studies, C.E. Shannon and J. McCarthy, eds., p. 43–98, Princeton, NJ: Princeton University Press, 1956.

1105. Wahba G. Spline Models for Observational Data, SIAM, 1990.

1106. Wahba G., D.R. Johnson, F. Gao and J. Gong. "Adaptive tuning of numerical weather prediction models: Randomized GCV in three and four dimensional data assimilation", Monthly Weather Review, 1995, vol. 123, p. 3358–3369.

1107. Waibel A., T. Hanazawa, G. Hinton, K. Shikano and K.J. Lang. "Phoneme recognition using time-delay neural networks", IEEE Transactions on Acoustics, Speech and Signal Processing, 1989, vol. ASSP-37, p. 328–339.

1108. Waltz D. "Neural nets and AI: Time for a synthesis", plenary talk, International Conference on Neural Networks, 1997, vol. 1, p. xiii, Houston.

1109. Waltz M.D. and K.S. Fu. "A heuristic aproach to reinforcement learning control systems", IEEE Transactions on Automatic Control, 1965, vol. AC-10, p. 390-398.

1110. Wan E.A. "Time series prediction by using a connectionist network with internal delay lines", in Time Series Prediction: Forecasting the Future and Understanding the Past, A.S. Weigend and N.A. Gershenfield, eds., p. 195–217. Reading, MA: Addison-Wesley, 1994.

1111. Wan E.A. "Temporal backpropagation for FIR neural networks", IEEE International Joint Conference on Neural Networks, 1990, vol. I, p. 575–580, San Diego, CA.

1112. Wan E.A. and F. Beaufays. "Diagrammatic derivation of gradient algorithms for neural networks", Neural Computation, 1996, vol. 8, p. 182–201.

1113. Watanabe H., Yamaguchi and S. Katagiri. "Discriminative metric design for robust pattern recognition", IEEE Transactions on Signal Processing, 1997, vol. 45, p. 2655–2662.

1114. Waterhouse S., D. MacKay and A. Robinson. "Bayesian methods for mixtures of experts", Advances in Neural Information Processing Systems, 1996, vol. 8, p. 351–357, Cambridge, MA: MIT Press.

1115. Watkins C.J.C.H. Learning from Delayed Rewards, Ph.D. Thesis, University of Cambridge, England, 1989.

1116. Watkins C.J.C.H. and P. Dayan. "Q-learning", Machine Learning, 1992, vol. 8, p. 279–292.

1117. Watrous R.L. "Learning algorithms for connectionist networks: Aplied gradient methods of nonlinear optimization", First IEEE Infemational Conference on Neural Networks. 1987, vol. 2, p. 619–627, San Diego, CA.

1118. Watson G.S. "Smooth regression analysis", Sankhya: The Indian Journal of Statistics, Series A, 1964, vol. 26, p. 359–372.

1119. Webb A.R. "Functional aproximation by feed-forward networks: A least-squares aproach to generalisation", IEEE Transactions on Neural Networks, 1994, vol. 5, p. 480–488.

1120. Webb A.R. and D. Lowe. "The optimal internal representation of multilayer classifier networks performs nonlinear discriminant analysis", Neural Networks, 1990, vol. 3, p. 367–375.

1121. Weigend A.S., B. Huberman and D. Rumelhart. "Predicting the future: A connectionist aproach", International Journal of Neural Systems, 1990, vol. 3, p. 193–209.

1122. Weigend A.S., D.E. Rumelhart and B.A. Huberman. "Generalization by weight-elimination with aplication to forecasting", Advances in Neural Information Processing Systems, 1991, vol. 3, p. 875–882, San Mateo, CA: Morgan Kaufmann.

1123. Weigend A.S. and N.A. Gershenfield, eds. Time Series Prediction: Forecasting the Future and Understanding the Past, vol. 15, Santa Fe Institute Studies in the Sciences of Complexity, Reading, MA: Addison-Wesley, 1994.

1124. Weierstrass K. "Uber die analytische Darstellbarkeit sogenannter willkurlicher Funktionen einer reellen veranderlichen", Sitzungsberichte der Akademie der Wissenschaften, Berlin, 1885, p. 633–639, 789–905.

1125. Werbos P.J. "Neural networks and the human mind: New mathematics fits humanistic insight", IEEE International Conference on Systems, Man and Cybernetics, 1992, vol. 1, p. 78–83, Chicago.

1126. Werbos P.J. "Backpropagation through time: What it does and how to do it", Proceedings of the IEEE, 1990, vol. 78, p. 1550–1560.

1127. Werbos P.J. "Backpropagation and neurocontrol: A review and prospectus", International Joint Conference on Neural Networks, 1989, vol. I, p. 209–216, Washington, DC.

1128. Werbos P.J. "Beyond regression: New tools for prediction and analysis in the behavioral sciences", Ph.D. Thesis, Harvard University, Cambridge, MA, 1974.

1129. Wettschereck D. and T. Dietterich. "Improving the performance of radial basis function networks by learning center locations", Advances in Neural Information Processing Systems, 1992, vol. 4, p. 1133–1140, San Mateo, CA: Morgan Kaufmann.

1130. White D.A. and D.A. Sofge, eds. Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Aproaches, New York: Van Nostrand Reinhold, 1992.

1131. White H. Artificial Neural Networks: Aproximation and Learning Theory, Cambridge, MA: Blackwell, 1992.

1132. White H. "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mapings", Neural Networks, 1990, vol. 3, p. 535–549.

1133. White H. "Learning in artificial neural networks: A statistical perspective", Neural Computation, 1989, vol. 1, p. 425–464.

1134. White H. "Some asymptotic results for learning in single hidden-layer feedforward network models", Journal of the American Statistical Society, 1989Б, vol. 84, p. 1003–1013.

1135. Whitney H. "Differentiable manifolds", Annals of Mathematics, 1936, vol. 37, p. 645–680.

1136. Whittaker E.T. "On a new method of graduation", Proceedings of the Edinburgh Mathematical Society, 1923, vol. 41, p. 63–75.

1137. Widrow B. "Generalization and information storage in networks of adeline 'neurons"', in M.C. Yovitz, G.T. Jacobi and G.D., Goldstein, eds., Self-Organizing Systems, p. 435–461, Washington, DC: Spartan Books, 1962.

1138. Widrow B., J.M. McCool, M.G. Larimore and C.R. Johnson, Jr. "Stationary and nonstationary learning characteristics of the LMS adaptive filter", Proceedings of the IEEE, 1976, vol. 64, p. 1151–1162.

1139. Widrow B., J.R. Glover, Jr., J.M. McCool, J. Kaunitz, C.S. Williams, R.H. Hearn, J.R. Zeidler, J. Dong, Jr. and R.C. Goodlin. "Adaptive noise cancelling: Principles and aplications", Proceedings of the IEEE, 1975, vol. 63, p. 1692–1716.

1140. Widrow B., N.K. Gupta and S. Maitra. "Punish/reward: Learning with a critic in adaptive threshold systems", IEEE Transactions of Systems, Man and Cybernetics, 1973, vol. SMC-3, p. 455–465.

1141. Widrow B. and M.E. Hoff, Jr. "Adaptive switching circuits", IRE WESCON Convention Record, 1960, p. 96–104.

1142. Widrow B. and M.A. Lehr. "30 years of adaptive neural networks: Perceptron, madaline and backpropagation", Proceedings of the Institute of Electrical and Electronics Engineers, 1990, vol. 78, p. 1415–1442.

1143. Widrow B., P.E. Mantey, L.J. Griffiths and B.B. Goode. "Adaptive antenna systems", Proceedings of the IEEE, 1967, vol. 55, p. 2143–2159.

1144. Widrow B. and S.D. Stearns. Adaptive Signal Processing, Englewood Cliffs, NJ: Prentice-Hall, 1985.

1145. Widrow B. and E. Walach. Adaptive Inverse Control, Uper Saddle River, NJ: Prentice-Hall, 1996.

1146. Wieland A. and R. Leighton. "Geometric analysis of neural network capabilities", First IEEE International Conference on Neural Networks, 1987, vol. III, p. 385–392, San Diego, CA.

1147. Wiener N. Cybernetics, 2nd edition New York: Wiley, 1961.

1148. Wiener N. Nonlinear Problems in Random Theory, New York: Wiley, 1958.

1149. Wiener N. Extrapolation, Interpolation and Smoothing of Stationary Time Series with Engineering Aplications, Cambridge, MA: MIT Press, 1949. (Ранее было опубликовано как отчет с результатами исследований Службы национальной безопасности; февраль, 1942).

1150. Wiener N. Cybernetics: On Control and Communication in the Animal and the Machine, New York: Wiley, 1948.

1151. Wilks S.S. Mathematical Statistics, New York: Wiley, 1962.

1152. Williams R.J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning", Machine Learning, 1992, vol. 8, p. 229–256.

1153. Williams R.J. "Toward a theory of reinforcement-learning connectionist systems", Technical Report NU-CCS-88-3, College of Computer Science, Northeastern University, Boston, 1988.

1154. Williams R.J. "Feature discovery through error-correction learning", Technical Report ICS-8501. University of California, San Diego, CA, 1985.

1155. Williams R.J. and J. Peng. "An efficient gradient-based algorithm for on-line training of recurrent network trajectories", Neural Computation, 1990, vol. 2, p. 490–501.

1156. Williams R.J. and D. Zipser. "Gradient-based learning algorithms for recurrent networks and their computational complexity", in Y. Chauvin and D.E. Rumelhart, eds., Backpropagation: Theory, Architectures and Aplications, p. 433–486, Hillsdale, NJ: Lawrence Erlbaum, 1995.

1157. Williams R.J. and D. Zipser. "A learning algorithm for continually running fully recurrent neural networks", Neural Computation, 1989, vol. 1, p. 270–280.

1158. Willshaw D.J., O.P. Buneman and H.C. Longuet-Higgins. "Non-holographic associative memory", Nature (London), 1969, vol. 222, p. 960–962.

1159. Willshaw D.J. and C. von der Malsburg. "How patterned neural connections can be set up by self-organization", Proceedings of the Royal Society of London Series B, 1976, vol. 194, p. 431–445.

1160. Wilson G.V. and G.S. Pawley. "On the stability of the travelling salesman problem algorithm of Hopfield and Tank", Biological Cybernetics, 1988, vol. 58, p. 63–70.

1161. Wilson H.R. and J.D. Gowan. "Excitatory and inhibitory interactions in localized populations of model neurons", Journal of Biophysics, 1972, vol. 12, p. 1–24.

1162. Winder R.O.x. "Single stage threshold logic", Switching Circuit Theory and Logical Design, AIEE Special Publications, 1972, vol. S-134, p. 321–332.

1163. Winograd S. and J.D. Cowan. Reliable Computation in the Presence of Noise, Cambridge, MA: MIT Press, 1963.

1164. Wolpert D.H. "Stacked generalization", Neural Networks, 1992, vol. 5, p. 241–259.

1165. Wood N.L. and N. Cowan. "The cocktail party phenomenon revisited: Attention and memory in the classic selective listening procedure of Cherry (1953)", Journal of Experimental Psychology: General, 1995, vol. 124, p. 243–262.

1166. Woods W.A. "Important issues in knowledge representation", Proceedings of the Institute of Electrical and Electronics Engineers, 1986, vol. 74, p. 1322–1334.

1167. Wu C.F.J. "On the convergence properties of the EM algorithm", Annals of Statistics, 1983, vol. 11, p. 95–103.

1168. Wylie C.R. and L.C. Barrett. Advanced Engineering Mathematics, 5th edition, New York: McGrawHill, 1982.

1169. Xu L., A. Krzyzak and A. Yuille. "On radial basis function nets and kernel regression: Statistical consistency, convergency rates and receptive field size", Neural Networks, 1994, vol. 7, p. 609–628.

1170. Xu L., E. Oja and C.Y. Suen. "Modified Hebbian learning for curve and surface fitting", Neural Nehvorks, 1992, vol. 5, p. 441–457.

1171. Yang H. and S. Amari. "Adaptive online learning algorithms for blind separation: Maximum entropy and minimum mutual information", Neural Computation, 1997, vol. 9, p. 1457–1482.

1172. Yee P.V. Regularized Radial Basis Function Networks: Theory and Aplications to Probability Estimation, Classification and Time Series Prediction, Ph.D. Thesis, McMaster University, Hamilton, Ontario, 1998.

1173. Yockey H.P. Information Theory and Molecular Biology, Cambridge: Cambridge University Press, 1992.

1174. Yoshizawa S., M. Morita and S. Amari. "Capacity of associative memory using a nonmonotonic neuron model", Neural Networks, 1993, vol. 6, p. 167–176.

1175. Zadeh L.A. "Outline of a new aproach to the analysis of complex systems and decision processes", IEEE Transactions on Systems, Man and Cybernetics, 1973, vol. SMC-3, p. 28–44.

1176. Zadeh L.A. "Fuzzy sets", Information and Control, 1965, vol. 8, p. 338–353.

1177. Zadeh L.A. "A contribution to the theory of nonlinear systems", J. Franklin Institute, 1953, vol. 255, p. 387–401.

1178. Zadeh L.A. and C.A. Desoer. Linear System Theory: The State Space Aproach, New York: McGraw-Hill, 1963.

1179. Zames G. "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms and aproximate inverses", IEEE Transactions on Automatic Control, 1981, vol. AC-26, p. 301–320.

1180. Zames G. and B.A. Francis. "Feedback, minimax, sensitivity and optimal robustness", IEEE Transactions on Automatic Control, 1983, vol. AC-28, p. 585–601.

1181. Zeevi A.J., R. Meir and V. Majorov. "Error bounds for functional aproximation and estimation using mixtures of experts", IEEE Transactions on Information Theory, 1998, vol. 44, p. 1010–1025.

1182. Zeki S. A Vision of the Brain, Oxford: Blackwell Scientific Publications, 1993.

1183. Zipser D. and D.E. Rumelhart. "The neurobiological significance of the new learning models", in Computational Neuroscience, E.L. Schwartz, ed., p. 192–200, Cambridge, MA: MIT Press, 1990.

Предметный указатель

A

A-conjugate, *319*
Accuracy of
  empirical fit to the approximation, *285*
  the best approximation, *285*
Action potential, *39*
Activation
  function, *43*; *247*
  potential, *44*
Activity, *511*
  pattern, *909*
  product rule, *98*
AdaBoost, *470*
Adaptive
  filter, *173*; *175*
  pattern classification, *606*; *990*
  principal components extraction, *547*
  process, *176*
  resonance theory, *79*
Affine transformation, *43*; *374*
Agent, *762*
AI, *71*
Anatomical connectivity, *909*
Annealing schedule, *709*
Anti-Hebbian learning rule, *550*
APEX, *547*; *558*
Approximate
  cost-to-go function, *778*
  data analysis, *688*
  policy iteration algorithm, *780*
Approximation, *46*
  error, *139*
    vector, *521*
  problem, *142*
AR-процесс, *67*
Artificial intelligence, *71*
Associated perfectly, *131*
Association probability, *744*
Associative
  Gaussian mixture model, *460*; *482*
  memory, *77*; *111*
Asymptotic
  mode, *294*
  stability theorem, *529*
Attentional network, *994*
Autoassociation, *111*
Average operator, *137*
Averaging, *206*

B

Back propagation, *81*
  algorithm, *82*; *220*
  formula, *231*
  learning, *220*
  through time, *981*
Backward computation, *243*
Barlow's hypothesis, *647*
Batch
  mode, *238*
  SOM, *593*
Bayes
  classifier, *206*
  probability error, *95*
Bellman's optimality equation, *770*
Bernoulli trial, *348*; *698*
Best-approximation property, *371*
Between-class covariance matrix, *273*
Bias, *43*; *139*; *579*
Bias/variance dilemma, *140*
Bit, *625*
Blind
  deconvolution, *685*
  equalization, *686*
  signal separation, *119*
  source separation, *596*
    problem, *623*; *657*
Block diagram, *49*
Boltzmann
  factor, *693*
  learning, *163*
    rule, *105*; *720*
  machine, *81*; *104*; *691*; *713*

Boosting, *459*; *466*
  algorithm, *467*
BOSS, *940*
BPTT, *943*; *981*
Bracketing phase, *325*
Brain, *37*
Brent's method, *326*
BSB, *884*

C

Calculus of variations, *734*
Canonical
  correlation analysis, *652*
  distribution, *693*
  pair, *454*
Capacity, *147*
Center of expansion, *363*
Central
  limit theorem, *642*
  nervous system, *40*
Chain rule, *227*; *681*
  of calculus, *951*; *971*
Channel capacity, *642*; *648*
Chaos, *83*
Chapmen-Kolmogorov identity, *697*
Chemical synapse, *38*
Cholesky factorization, *960*
Clamped condition, *105*
Classification, *113*
  and regression tree, *486*
Closed-loop
  feedback, *93*
  operator, *53*
Clustering, *743*
  effect, *401*
Cocktail party problem, *119*
Code book, *603*
Coefficient of expansion, *363*
Cofactor, *665*
Cohen-Grossberg theorem, *882*
Coherent region, *612*
Combinatorial optimization, *710*
  problem, *909*
Committee machine, *458*
Communication
  link, *51*
  process, *622*
Community, *131*
Compact representation, *779*
Competition, *579*
Competitive learning, *79*; *101*; *163*; *573*
Complete data set, *496*
Complete-data log-likelihood function, *497*
Complexity
  penalty, *296*
  polynomial, *310*
  theory, *911*
Computation node, *51*
Computational complexity, *162*; *310*
Concept, *158*
  class, *158*
  formation, *908*
Condition number, *192*; *371*
Conditional
  differential entropy, *635*
  entropy, *632*
Confidence, *35*
  interval, *154*
  parameter, *159*
Conjugate-direction method, *319*
Conjugate-gradient method, *319*; *322*
Connecting link, *42*
Connectionist
  network, *33*
  paradigm, *306*
Connectivity, *220*; *511*
Consistence, *160*
Consistent estimator, *392*
Constrained optimization problem, *301*
Constraining, *333*
Context unit, *923*
Contextual
  information, *35*
  knowledge, *991*
Control, *72*
  mechanism, *91*
Convolution, *331*
  network, *65*

sum, *801*; *820*
Cooperation, *579*
Correlation
coefficient, *610*
dimension, *897*
function, *896*
matrix, *185*
memory, *77*; *128*
Cost function, *91*; *226*; *821*
Cost-to-go function, *765*
Covariance hypothesis, *98*
Credit assignment, *106*
problem, *79*; *229*
Cross-correlation, *542*
vector, *185*
Cross-validation, *289*
Curse of dimensionality, *286*; *778*
Curve-fitting, *278*
procedure, *325*

D

Darwinian learning model, *163*
Data, *72*
reconstruction, *543*
space, *514*
Decision stump, *484*
Decoupled extended Kalman filter (DEKF), *963*; *981*
Delay-embedding theorem, *899*
Delayed reinforcement, *109*
Delta rule, *173*; *228*
Dendrite, *38*
Dendritic
subunit, *39*
tree, *39*
Density estimation, *390*
Desired response, *33*; *59*; *91*; *176*
vector, *351*
Deterministic
annealing, *742*
approximation, *730*
Boltzmann learning rule, *733*
finite-state automata, *926*
ordinary differential equation, *528*
DFA, *926*
DFP, *329*
Diagonal map, *679*
Dichotomy, *147*
Differential entropy, *627*
Digit recognition problem, *60*
Dimensionality
of the input space, *174*
reduction, *514*; *520*; *600*
Direct
learning, *118*
problem, *353*
Discount factor, *764*
Distortion measure, *589*; *743*
Distributed
memory, *123*
time lagged feedforward network, *800*
Dual problem, *424*
Duality theorem, *424*
Dynamic
programming, *110*; *761*
structure, *459*
Dynamical system, *174*

E

Early stopping method of training, *292*
EDVAC, *76*
Effective
input, *526*
width, *581*
Eigen-analyzer, *543*
Eigenanalysis, *871*
Eigencomposition, *519*
Eigenfunction, *436*
Eigenvalue, *436*; *518*
problem, *518*
Eigenvector, *518*
Emergent, *865*
Empirical risk, *284*
functional, *254*
minimization, *144*
Enhancement of polarization target, *653*
ENIAC, *76*
Ensemble averaging, *459*; *460*

Entropy, *625*
Epoch, *226*; *261*
Equilibrum state, *931*
Ergodicity, *699*
  theorem, *701*
Error, *67*; *112*
  back-propagation algorithm, *219*
  covariance matrix, *959*
  parameter, *159*
  signal, *91*; *107*; *219*; *223*
  surface, *107*
Error-correction learning, *91*; *163*
  rule, *206*; *219*
Estimation
  error, *139*
  learning curve, *293*
  subset, *289*
Euler-Lagrange equation, *361*
Exclusive OR problem, *243*
Exhaustive learning, *294*
Existence theorem, *283*
Expectation-maximization approach, *496*
Expectational error, *135*
Expected distortion, *589*; *743*
Experience, *127*
Expert
  network, *479*
  system, *73*
Explanation level, *73*
External approximation, *300*
Externally recurrent network, *980*

F

Factorial
  code, *647*
  distribution, *637*; *737*
Feature
  detector, *101*; *268*
  extraction, *514*
  map, *331*; *588*
  selection, *514*; *605*
  space, *268*; *344*; *433*; *514*
  vector, *434*
Feature-mapping model, *575*
Feedback, *52*; *835*
  control system, *117*; *977*
  factor, *884*
  loop, *57*; *176*
Feedforward
  connection, *549*
  net, *56*
Filtered estimation error, *959*
Filtering, *118*
  process, *175*
Finite-state
  automata, *937*
  machine, *941*
Finite-time approximation, *709*
FIR-фильтр, *815*
First absolute moment, *284*
First-order autoregressive model, *609*
Fisher's
  criterion, *273*
Fitting
  number, *290*
  procedure, *350*
Fletcher-Reeves formula, *323*
Focused time lagged feedforward network, *800*
Focusing, *991*
Forgetting, *527*
  term, *583*
Forward
  computation, *242*
  pass, *219*; *232*
Fractal dimension, *894*
Function
  signal, *223*
  space, *356*
Future extraction, *113*
Fuzzy
  system, *992*

G

Galerkin's method, *371*
Gamma memory, *804*
Gating network, *459*; *479*
Gaussian
  classifier, *402*

function, *352*
GCV, *908*
GEKF, *963*
Generalization, *33*; *60*
error, *155*
phase, *350*
Generalized
cross validation, *383*; *908*
delta rule, *236*
Hebbian algorithm, *538*
Lloyd algorithm, *590*
Generative model, *696*; *728*
parameter vector, *479*
GHA, *538*
Gibbs
distribution, *691*
sampler, *711*
Global
extended Kalman filter, *963*
feature, *287*
feedback, *981*
rejection index, *674*
Gradient
descent, *228*
matrix, *952*
Green's
identity, *360*
matrix, *365*
Growth function, *148*
GSLC, *122*
GVC, *383*

H

Heaviside, *896*
function, *45*
Hebb's postulate of learning, *95*
Hebbian
learning rule, *549*; *721*
synapse, *96*
Helmholtz machine, *728*
Hessian matrix, *276*
Heteroassociation, *111*
Heuristic reinforcement signal, *109*
Hidden
layer, *219*
neuron, *56*; *268*
space, *268*; *344*
unit, *56*
Hierarchical
clustering algorithm, *570*
mixture of experts, *460*; *484*
Hippocampus, *100*
HME-структура, *460*
Hold out method, *294*
Hopfield network, *79*; *80*
Horizontal cell, *37*
Human vision, *31*
Hybrid
learning process, *399*
system, *75*
Hyperbolic tangent, *247*
Hypothesis
boosting problem, *467*
space, *159*
testing procedure, *206*

I

Image
coding, *544*
processing, *331*
Implicit representation, *799*
Impulse response, *802*
Independence theory, *190*
Independent
component analysis , *658*
variable, *135*
Indicator variable, *499*
Indirect
control, *979*
learning, *118*
Induced local field, *44*; *196*
Inference
nonparametric, *34*
statistical, *34*
Infimum, *143*
Infomax, *82*; *163*; *641*
Information
latching, *970*

preservation rule, *485*
theory, *622*
Information-preserving learning procedure, *965*
Information-theoretic model, *622*
Initial condition, *953*
Initialization, *241*; *251*
Inner-product kernel, *433*; *435*; *562*
Innovation process, *958*
Input
layer, *56*; *509*
signal, *59*
Input-output mapping, *33*
Instantaneous estimate, *108*; *312*
Interregional circuit, *39*
Invariance by
structure, *66*
training, *66*
Invariant feature, *66*
Inverse
function theorem, *933*
kinematics, *288*
parabolic interpolation, *326*
problem, *288*; *353*
Ising model, *80*
Isochronous correspondence, *845*
Iteration, *243*
Iterative adaptive scheme, *712*

J

Jensen's inequality, *507*; *735*
Joint
a posteriori probability, *491*
entropy, *633*

K

k-means
clasterization algorithm, *399*
k-nearest neighbor classifier, *95*
Kalman
filtering, *956*
algorithm, *960*
gain, *959*
Kernel regression, *390*
Key
matrix, *128*
pattern, *112*
Knowledge
base, *72*
representation, *58*
Kuhn-Tucker condition, *423*
Kullback-Leibler divergence, *626*

L

Lagrange multiplier, *301*; *630*
Lagrangian, *301*
Largest eigenvalue, *190*
Lateral
connection, *550*
inhibition, *101*
interaction, *580*
Lattice, *573*
spin, *79*
Law of large numbers, *145*; *152*
Learning, *32*; *72*
algorithm, *32*
curve, *192*
element, *72*
machine, *110*; *141*
matrix, *77*
paradigm, *90*
process, *123*
rate, *253*
parameter, *92*; *720*
system, *762*
theory, *140*
time, *277*
vector quantization (LVQ), *604*
without a teacher, *108*
Learning-rate parameter, *178*; *584*
Least-mean-square algorithm, *173*
Least-squares estimate, *543*
Left singular vector, *561*
Likelihood
equation, *493*
function, *493*; *718*
ratio, *208*; *257*
Line search, *325*
Linear

adaptive
filter, *186*
filtering, *173*
combiner, *46*; *121*
output, *43*
differential operator, *356*
least-squares filter, *184*
machine, *417*
Linearly
independent vector, *320*
separable
class, *200*
pattern, *78*
Linsker's model, *513*
LMS, *78*; *186*; *277*
Load parameter, *873*
Loading problem, *106*
Local
averaging, *331*
circuit, *39*
controllability theorem, *934*
feature, *287*
gradient, *228*
minima, *313*
observability theorem, *935*
receptive field, *140*
stability, *843*
Log-likelihood
equation, *494*
ratio test, *209*
Logistic
belief net, *722*
function, *46*; *220*
Look-up table, *609*; *787*
Loss function, *418*; *445*
Low-pass filter, *187*
Lyapunov
dimension, *898*
exponent, *897*

M

M-ary pulse-amplitude modulation, *621*
M-class classification problem, *253*
M-мерная амплитудно-импульсная модуляция, *621*
Madaline, *78*
Magnification factor, *594*
Mahalanobis distance, *62*
Majority rule, *880*
Mapping input-output, *33*
Margin of separation, *419*
Marginal
density, *712*
entropy, *638*
mismatch, *677*
Markov
blanket, *739*
chain, *692*; *696*
Matched filter, *537*
Matrix inversion lemma, *304*
Maximizing information content, *246*
Maximum
eigenfilter, *530*
energy gain, *311*
entropy
method, *678*
principle, *629*
likelihood estimate, *494*
mutual information, *82*; *641*
principle, *622*
ME-структура, *460*
Mean recurrence time, *699*
Mean-field
approximation, *732*
distribution, *735*
equation, *739*
theory, *692*
Measure of memory, *187*
Measurement
equation, *928*
matrix, *957*
Memorized
matrix, *128*
pattern, *112*
Memorizing, *279*
Memory, *122*; *800*
matrix, *126*
resolution, *803*

Memory-based learning, *93*; *163*
Mental
  process, *74*
  representation, *74*
Mercer's theorem, *435*
Message, *624*
Metastable state, *586*
Method of
  false nearest neighbors, *901*
  Lagrange multipliers, *423*
  potential functions, *83*
Metropolis, *704*
MIMO, *115*
Minimax criterion, *191*
Mixed nonstationary policy, *789*
Mixture of experts, *459*; *479*
MLP, *260*
Model
  order, *67*
  reference adaptive control, *977*
Modular network, *459*
Monomial, *344*; *439*
Most plausible value, *494*
MRAC, *977*
Multifold cross-validation, *294*
Multilayer perceptron, *219*; *260*
Multinomial probability, *481*
Multistage hierarchical vector quantizer, *608*
Multivariable interpolation, *350*
Mutual information, *633*
  principle, *163*

N

N-мерный единичный куб, *884*
NARX, *921*; *937*
Nat, *625*; *906*
Natural
  distribution, *895*
  gradient, *669*
Nearest neighbor rule, *94*; *590*
Nearest-neighbor quantizer, *603*
Negative
  binomial distribution, *348*
  example, *158*
Neighbourhood function, *582*
Nerve net, *37*
NETtalk, *302*; *403*
Network
  dimensionality, *125*
  of networks, *908*
  pruning, *296*
Neural
  image, *36*
  microcircuit, *39*
Neuro-beamformer, *122*
Neurobiological analogy, *36*
Neurocontroller, *117*
Neurodynamic programming, *761*
Neuromorphic integrated circuit, *37*
Newton direction, *327*
Newton's method, *180*; *318*
Neyman-Pearson criterion, *63*
Nonasymptotic mode, *294*
Nonlinear
  activation function, *220*
  adaptive filter, *77*; *134*
  feature detector, *256*
  input-output mapping, *281*
  regression problem, *378*
  regressive model, *446*
  stochastic difference equation, *527*
Nonreciprocal two-port device, *38*
Nonstationary environment, *34*; *562*
Norm weight matrix, *374*
Normalization, *525*
Normalized
  weighting function, *393*
  degree of freedom, *583*
  eigenvector, *542*
Normed space, *356*
NP-complete, *910*
NRWE, *404*
Numerical optimizing, *317*

O

OBD, *301*
OBS , *301*
Occam's razor, *279*; *474*

ODE, *528*
Off-line training, *966*
One-from-M coding scheme, *272*
Open-loop operator, *53*
Optical character recognition (OCR), *469*
Optimal
  brain
    damage, *301*
    surgeon, *301*
  hyperplane, *419*
  robust estimation procedure, *445*
Order, *928*
  of model, *923*
Ordinary cross-validation estimate, *382*
Orhtogonal vector, *130*
Orthogonal
  set, *131*
  similarity transformation, *519*
Outer product, *127*
  rule, *128*
Outlier, *95*
Output
  layer, *56*; *219*; *509*
  signal, *91*
Overdetermined problem, *156*
Overfitting, *292*; *353*; *460*; *474*

P

PAC, *157*; *467*
Parallel
  distributed processing, *313*
  identification model, *976*
Parallelism, *74*
Parity function, *314*
Partition function, *693*; *716*
Pattern classification, *34*
Peano
  curve, *599*
  string, *622*
Perceptron, *196*
  convergence
    algorithm, *197*; *204*
    theorem, *78*; *173*
Performance
  element, *72*
  index, *91*
  measure, *296*; *334*
Polak-Ribiere formula, *323*
Policy, *764*
  iteration algorithm, *771*
Polynomial time algorithm, *162*
Positive
  definite matrix, *181*
  example, *158*
Potentiation, *99*
Preprocessor, *68*; *569*
Primary
  reinforcement signal, *109*
  visual cortex, *616*
Principal
  component, *520*
  components analysis, *510*; *658*
  curve, *597*
  mode, *531*
  subspace, *559*
Principle of
  detailed balance, *704*
  minimal free energy, *695*
  minimum redundancy, *647*
  orthogonality, *136*; *522*
Probabilistic
  generative model, *492*
  model, *80*
  reasoning, *72*
Probability
  mass function, *626*
  of classification error, *152*
  of correct classification, *260*
  of detection, *63*
  of error, *260*
  of false alarm, *63*
Probably-approximately correct model, *158*
Process
  equation, *928*
  noice, *965*
Processing
  noise, *634*
  style, *73*

Prototype, *253*
  state, *864*
Pseudo-differential operator, *368*
Pseudoinverse method, *397*
Pseudotemperature, *105*
Pyramidal cell, *38*

Q

Q-factor, *770*
Q-обучение, *784*
Quadratic
  approximation, *300*; *318*
  function, *319*

R

Radial basis function (RBF), *82*; *341*; *342*; *350*; *374*
Rank deficient, *276*
Rate
  distortion theory, *607*
  of convergence, *193*
  of learning, *92*; *98*
RBF-сеть, *22*; *342*
Real-time recurrent learning, *949*; *981*
Receptive field, *64*; *331*; *375*; *513*
Recognition model, *728*
Recurrent, *919*
  multilayer perceptron, *925*; *967*
  network, *52*; *57*
Redundancy, *512*
Redundant data, *239*
Reestimation algorithm, *559*
Reestimator, *559*
Reference
  measure, *626*
  model, *979*
  signal, *117*
Reflection coefficient, *67*
Regression model, *135*
Regularization, *355*
  network, *369*
  parameter, *297*; *357*
  principle, *358*
  term, *356*
Reinforcement learning, *81*; *109*; *760*
Relative
  entropy, *626*
  gradient, *669*
Reliable estimate, *900*
Renormalized SOM algorithm, *621*
Repertoire, *164*
Replicator, *307*
Residual error, *322*
Response, *129*
Retraining, *134*
Ricatti equation, *960*
Risk functional, *143*
RMLP, *925*; *967*
Rozenblatt perceptron, *172*
RTRL, *981*

S

Saddle point, *251*; *423*
Sample complexity, *160*
  problem, *281*
Sampling density, *286*
Saturation, *525*
Scaling, *314*
  factor, *403*
Search time constant, *195*
Searching problem, *72*
Sectioning phase, *325*
Selective learning model, *163*
Self-adjoint operator, *365*
Self-amplification, *549*
Self-organized learning, *110*; *509*
  stage, *399*
Self-organizing map, *79*; *400*; *573*; *577*
Sensitivity, *275*; *311*
  factor, *227*
  graph, *241*; *276*
Separability of patterns, *343*
Separating
  capacity, *349*
  surface, *344*
Separatrix, *849*
Sequential
  mode, *238*

processing, *73*
Shift invariance, *331*
Short-range exdcitatory mechanism, *576*
Sigmoid
belief network, *81*; *692*; *722*
Sigmoidal nonlinearity, *220*; *234*
Signal
component, *536*
processing, *173*
vector, *91*
Signal-blocking matrix, *122*
Signal-to-noise ratio, *643*
SIMO, *803*
Simple recurrent network, *923*
Simulated annealing, *81*; *708*
Single input single output, *931*
Single-layer network, *56*
Single-loop feedback system, *52*
Singular
point, *842*
value decomposition, *560*
Singular-value decomposition, *398*
SISO, *931*
Slack variable, *447*
Slope parameter, *47*
Smoothing, *118*
parameter, *413*
Smoothness index, *386*
SOM, *588*
Sparse architecture, *991*
Spatial
difference, *652*
diversity, *657*
Spatially continous space, *588*
Spectral theorem, *519*
Spectrum, *70*
Spin-glass state, *880*
Spline, *288*
Spurious attractor, *871*
SRN, *923*
Stability-plastycity dilemma, *34*
Stabilized effect, *237*
Stabilizer, *356*
Stable, *54*
state, *867*
Stagecoach problem, *775*
Standard error term, *355*
State, *762*
distribution vector, *700*
portrait, *839*
space, *695*; *928*
transition, *926*
vector, *957*
State-space
framework, *981*
model, *837*; *923*
Static structure, *458*
Statistical mechanics, *691*
Steady-state probability, *699*
Steepest descent, *669*
Stochastic
approximation, *195*; *312*; *341*
feedback system, *189*
gradient approach, *495*
matrix, *697*
relaxation, *708*
Storage
capacity, *112*; *873*
element, *93*
phase, *112*; *866*
Strict consistency, *144*
Strong learning model, *466*
Structural
mismatch, *677*
pattem recognition, *809*
Subsampling, *331*
Subspace decomposition, *523*
Supervised learning, *33*; *107*
stage, *399*
Support vector, *421*
machine, *83*; *417*
Supremum, *143*
SVD, *398*; *560*
SVM, *417*
Symbol structure, *71*
Symbolic code, *613*
Symmetric
model, *559*

synaptic connection, *713*
Synapse, *38*; *39*; *42*
Synaptic
adaptation, *579*
link, *49*
modification, *76*
System identification, *974*

T

Target
concept, *158*
signal, *176*
value, *248*
Taylor series, *300*
Teacher, *141*
Temperature of the system, *694*
Temporal
difference learning, *795*
pattern recognition, *809*
Terminal node, *486*
Test set, *289*
Theory of neural group selection, *163*
Thermal equilibrium, *105*
Threshold, *579*
function, *45*
of the test, *257*
Time
delay, *800*; *808*
lagged feedforward network, *967*
structure, *70*
Time-dependent mechanism, *96*
Time-frequency analysis, *993*
TLFN, *811*; *967*
Topographic map, *39*; *573*
Topological
neighbourhood, *581*
ordering, *584*
Topologically ordered computational map, *574*
Total covariance matrix, *271*
Trace, *190*
Training
data, *60*
error, *152*; *155*
phase, *350*
sample, *33*; *60*
set, *289*
Transfer function, *49*; *802*
Transient state, *698*
Translationally invariant, *367*
Transmitter substance, *38*
Travelling salesman problem, *752*; *909*
TSP, *909*
Turing machine, *939*

U

Unconstrained optimization problem, *177*
Underdeterminated, *156*
Uniformly sampled, *799*
Unit
delay operator, *801*
hypercube, *243*
square, *243*
vector, *515*
Unit-delay
element, *58*
operator, *53*; *93*; *526*
Unitary invariance property, *455*
Universal
approximation theorem, *282*; *386*
approximator, *310*
Unstable, *54*
Unsupervised
clustering algorithm, *889*
learning, *110*; *509*

V

Validation
learning curve, *293*
subset, *289*
Value iteration algorithm, *773*
Vanishing gradients problem, *849*
Vapnik-Chervonenkis dimension, *147*; *417*
Variance, *139*
probe, *516*
VC-измерение, *83*; *147*
Vector
projector, *871*
quantization, *603*

theory, *589*
Vector-coding algorithm, *577*
Virtual support vector, *454*
Viterbi algorithm, *991*
VLSI, *35*
VOR, *36*

W

Wake-sleep algorithm, *729*
Weak learning model, *467*
Weight, *42*
  decay, *297*
  elimination, *298*
  vector, *351*
Weight-sharing, *64*; *376*
Weighted
  average, *393*
  norm, *373*
Widrow's rule of thumb, *281*
Winner-takes-all, *101*
Winning neuron, *573*
Within-class covariance matrix, *273*

X

XOR, *243*

Z

Z-преобразование, *801*

А

Абсолютная
  температура, *693*
  энтропия, *627*
Автомат, *937*
  DFA, *926*
  конечный, *941*
Авторегрессионая модель первого порядка, *609*
Авторегрессионный процесс, *67*
Агент, *762*
Адаптивная
  классификация
    множеств, *606*
    образов, *990*
  модель, *80*
  система, *76*
Адаптивное управление по эталонной модели, *977*
Адаптивность, *34*
Адаптивный
  метод, *562*
  резонанс, *80*
  фильтр, *173*; *175*
Аддитивная модель, *80*
Аксон, *38*
Активационная связь, *50*
Активация нейрона, *77*
Алгоритм
  AdaBoost, *466*
  APEX, *558*
  CART, *487*
  EM, *460*
  GHA, *543*
  LMS, *173*; *186*
  Q-обучения, *786*
  SOM, *588*
    ренормализованный, *621*
  адаптивного усиления, *470*
  адаптивный, *560*
  Бройдена–Флетчера–Гольдфарба–Шанно, *329*
  Витерби, *991*
  векторного кодирования, *577*
  вероятностный, *312*
  вычислительно эффективный, *221*; *310*
  Девидона–Флетчера–Пауэла, *329*
  декорреляции, *559*
  детерминированного отжига, *746*
  динамического программирования, *767*
  EM, *497*
  засыпания-пробуждения, *729*
  иерархической кластеризации, *570*
  итерации по
    значениям, *773*
    стратегиям, *771*
  квантования векторов обучения, *605*
  кластеризации, *399*; *401*
    без учителя, *889*

по k-средним, *399*
линейного поиска, *325*
Метрополиса, *692*; *704*
минимизации среднеквадратической ошибки, *173*; *186*; *212*
на основе коррекции ошибок, *219*
наименьших квадратов, *78*
обратного распространения, *22*; *81*; *220*; *227*
во времени, *943*
ошибки, *82*; *219*; *464*
обучения, *32*; *90*
Хебба, *538*
повторного оценивания, *559*
с полиномиальным временем выполнения, *162*
слепого разделения источников, *677*
сопряженных градиентов, *327*
сходимости персептрона, *197*; *204*
усиления, *467*
фильтрации Калмана, *960*
эффективный, *162*
Анализ, *520*
временных рядов, *989*
главных компонентов, *23*; *510*; *515*; *574*
на основе ядра, *562*
канонической корреляции, *652*
независимых компонентов, *658*
собственных
значений, *543*
чисел, *871*
частотно- временной, *993*
Апостериорная вероятность, *255*
Аппроксимация, *46*
внешняя, *300*
глобальная, *390*
квадратичная, *300*; *318*
линейная, *317*
локальная, *390*
на конечном интервале времени, *709*
наилучшая, *371*
среднего поля, *732*
стохастическая, *312*; *341*
функций, *114*
Априорная
вероятность, *207*; *260*
информация, *59*
Архитектура, *274*
фон Неймана, *78*
Ассоциативная
вероятность, *744*
гауссова модель смешения, *460*; *482*
машина, *22*; *458*
память, *23*; *77*; *111*; *890*
Аттрактор, *848*
гиперболический, *849*
странный, *893*
точечный, *848*
Аттракторная нейронная сеть, *81*
Аффинное преобразование, *43*; *374*
Ацикличная сеть, *56*

Б

База знаний, *72*
Базис пространства данных, *520*
Байесовская
вероятность ошибки, *95*
Байесовский классификатор, *206*; *208*
Бассейн аттракции, *529*; *849*; *889*
Безусловная оптимизация, *177*
Белый шум, *536*
Бинарная классификация, *464*
Бинарное разбиение, *343*
Биномиальное распределение, *348*
Бит, *625*; *865*
Блок усреднения по ансамблю, *462*
Блокирование информации, *970*
Блочная диаграмма, *49*
Бритва Оккама, *279*; *474*

В

Вариационное исчисление, *734*
Вектор
активности, *909*
весов, *351*
взаимной корреляции, *185*
градиента, *177*
единичный, *515*

желаемого отклика, *351*
левый сингулярный, *398*; *561*
линейно-независимый, *320*
наблюдения, *957*
опорный, *421*
виртуальный, *454*
ортогональный, *130*
ошибки, *321*
аппроксимации, *521*
параметров порождающей модели, *479*
правый сингулярный, *398*; *561*
признаков, *434*
распределения состояний, *700*
сигнала, *91*
случайный, *515*
собственный, *523*
состояния, *837*; *957*; *959*
Векторное
квантование, *577*; *603*
кодирование, *577*
Векторный проектор, *871*
Вероятностная
модель, *80*
порождающая, *492*
функция массы, *626*
Вероятностное рассуждение, *72*
Вероятность
активации, *48*
апостериорная, *255*
априорная, *207*; *260*
ложной тревоги, *63*
обнаружения, *63*
ошибки, *260*
классификации, *152*
перехода, *696*; *697*; *706*
правильной классификации, *260*
Верхний предел, *143*
Ветвь, *49*
Взаимная
информация, *633*; *901*
корреляция, *542*
Винеровское решение, *186*
Вложенное подпространство, *484*
Внешне рекуррентная сеть, *980*
Внешнее произведение, *127*
Внешняя аппроксимация, *300*
Возбуждающий механизм близкого радиуса действия, *576*
Восстановление данных, *543*
Временная
задача, *106*; *110*
задержка, *800*
структура, *70*
Время, *799*
обучения, *277*
Входное пространство, *174*
Входной
сигнал, *59*
слой, *56*; *219*
Выбор разбиений, *487*
Выброс, *39*; *95*
Выделение признаков, *514*
Выходной
сигнал, *91*
слой, *56*; *219*
Вычислительная
мощность, *147*
сложность, *162*; *310*
Вычислительный узел, *51*

Г

Гамма-память, *804*
Гамма-функция, *805*
Гарантированный риск, *155*
Гессиан, *180*; *276*
Гибридная система, *75*
Гиперболический тангенс, *48*
Гиперплоскость, *344*
оптимальная, *419*
Гиперсфера, *345*
Гипокампус, *100*
Гипотеза
Барлоу, *647*
ковариации, *98*
Хебба, *583*
Чарча-Тьюринга, *939*
Главная
кривая, *597*

мода, *531*
Главный компонент, *520*
Глобальная обратная связь, *835*; *981*
Глобальное свойство, *56*
Глобальный
индекс отклонения, *674*
признак, *287*
расширенный фильтр Калмана, *963*
Глубина
памяти, *803*
усечения, *946*
Головка чтения-записи, *939*
Горизонт прогнозирования, *898*
Горизонтальная клетка, *37*
Градиент, *177*
локальный, *228*
поверхности ошибок, *108*
Граница
классификации, *474*
мягкая, *428*
разделения, *419*; *421*
решений, оптимальная, *258*
Граничная
плотность, *712*
энтропия, *638*
Граничное несоответствие, *677*
Граф
архитектурный, *51*
передачи сигнала, *49*
чувствительности, *241*; *276*; *955*
График ошибки, *192*
Грубость решения, *379*

Д

Данные, *72*
избыточные, *239*
Двоичная классификация, *147*
Двумерная решетка, *575*
Действие, *117*
Декодирование, *309*; *864*
Декомпозиция
на основе собственных векторов, *560*
Пифагора, *639*
пространственная, *523*
сингулярная, *398*; *560*
Дельта-правило, *91*; *173*; *228*; *239*
обобщенное, *236*; *243*
Дельта-распределение Дирака, *360*
Дельта-функция
Дирака, *592*
Дендритное дерево, *39*
Дендритный субблок, *39*
Дерево
классификации и регрессии, *486*
решений, *460*
мягких, *484*
Детектор признаков, *101*; *256*; *268*
Детерминированная функция, *135*
Детерминированное
правило обучения Больцмана, *733*
приближение, *730*
Детерминированный
конечный автомат, *926*
отжиг, *742*
Дешифратор, *521*
Диагональное отображение, *679*
Диаграмма перехода состояний, *701*
Дивергенция Кулбека–Лейблера, *626*; *636*
Дилемма
смещения/дисперсии, *140*; *358*; *461*
стабильности-пластичности, *34*
Динамическая
память, *924*
система, *174*
структура, *459*; *476*
Динамический подход, *134*
Динамическое
восстановление, *899*
программирование, *23*; *110*; *761*
Дисконтирующий множитель, *764*
Дискретная аппроксимация, *597*
Дискриминантная функция, *579*
Дисперсионный зонд, *516*
Дисперсия, *139*; *515*
Диссипативность, *850*
Диффеоморфизм, *899*; *933*
Дифференциал Фреше, *358*
Дифференциальная энтропия, *627*

Дихотомия, *147*; *343*
Доверительный интервал, *154*
Долговременная память, *123*
Долгосрочные зависимости, *973*
Достижимое состояние, *698*
Достоверность, *35*
  классификации, *474*

Е

Евклидово расстояние, *61*
Единичный
  вектор, *515*
  квадрат, *243*
ЕМ-алгоритм, *497*
Емкость, *147*
  канала, *642*
  памяти, *112*; *131*; *873*

Ж

Жадная стратегия, *771*
Желаемый
  выход, *91*
  отклик, *33*; *59*; *205*

З

Зависимая переменная, *135*
Задача
  XOR, *342*; *376*; *439*
  аппроксимации, *142*
    кривой, *278*; *341*
  безусловной оптимизации, *177*
  бинарной классификации, *464*
  временная, *106*; *110*
  двоичной классификации, *472*
  двойственная, *424*
  дилижанса, *775*
  загрузки, *106*
  избыточно определенная, *353*
  исключающего ИЛИ, *243*
  классификации, *474*
    на М классов, *253*
    точек единичного гиперкуба, *243*
  коммивояжера, *752*; *909*
  линейной адаптивной фильтрации, *173*
  максимизации, *495*
  назначения коэффициентов доверия, *229*
  недоопределенная, *156*
  неквадратичной оптимизации, *324*
  нелинейной регрессии, *378*
  обратная, *288*; *353*
  обращающихся в нуль градиентов, *849*
  определения собственных значений, *518*
  оптимальной фильтрации, *186*; *956*
  оптического распознавания символов, *469*
  оценки параметров, *570*
  переопределенная, *156*
  плохо обусловленная, *122*; *276*; *353*
  поиска, *72*
  прогнозирования, *957*
  прямая, *353*; *422*
  распознавания голоса, *119*
  расширения цели поляризации, *653*
  регрессии, *478*
  с неполными данными, *496*
  сверхопределенная, *373*; *377*
  сглаживания, *957*
  слепого восстановления сигнала, *685*
  слепого разделения источников, *657*
    сигнала, *623*
  сложности выборки, *281*
  структурная, *106*
  усиления гипотезы, *467*
  условной оптимизации, *301*; *422*
  фильтрации, *957*
  хорошо обусловленная, *353*
  что–где, *504*
Задержка
  по времени, *808*
  эхо-сигнала, *70*
Закон больших чисел, *145*; *152*; *699*
Запоминание, *279*
Зеркальное состояние, *880*
Знание, *58*
  эмпирическое, *135*
Значение
  сингулярное, *398*; *561*
  собственное, *190*; *436*; *518*; *523*; *543*
  стационарное, *516*

установившееся, *544*
экстремальное, *516*
Зонд, *867*
Зрительная кора мозга, *616*

И

Идентификация систем, *115*; *974*
Иерархическое
векторное квантование, *609*
смешение мнений экспертов, *22*; *460*; *484*
Избыточное
обучение, *279*; *460*; *473*
представление, *78*
Избыточность, *512*
Извлечение
главных компонентов, адаптивное, *547*
признаков, *113*; *331*; *514*; *605*
Измерение
Вапника–Червоненкиса, *22*; *83*; *147*; *417*
корреляции, *897*
Ляпунова, *898*
Изохронное соответствие, *845*
Инвариантность
Доплера, *68*
к поворотам, *367*
к преобразованиям, *367*
к смещению, *331*
по обучению, *66*
структурная, *66*
Инвариантный признак, *66*
Инверсная система, *116*
Индекс
гладкости, *386*
производительности, *91*
Индуцированное локальное поле, *44*; *196*; *225*
Инициализация, *241*; *251*
Интеллект, *990*
Интеллектуальная
машина, *23*
Интеллектуальное управление, *992*
Интерактивность, *96*
Интерполяция
обратная параболическая, *326*
строгая, *367*
функции многих переменных, *350*
Информационно-теоретическая модель, *622*
Информация
априорная, *59*
взаимная, *901*
контекстная, *35*
неявная, *72*
текстурная, *546*
явная, *72*
Искривление, *592*
Искусственная
нейронная сеть, *22*
топографическая карта, *575*
Искусственный интеллект, *71*
Итеративная адаптивная схема, *712*
Итерация, *224*; *243*
по стратегиям, *770*

К

Каноническая пара, *454*
Каноническое распределение, *693*
Карта
идентичности, *307*
признаков, *331*; *588*
самоорганизации, *23*; *79*; *400*; *573*
Квадратичная
аппроксимация, *300*; *318*
функция потерь, *142*
Квадрика, *344*
Квантизатор Гиббса, *711*
Квантование
вектора обучения, *574*
Квантователь Вороного, *603*
Класс
коррелированный, *209*
линейно-разделимый, *200*
понятий, *158*
Классификатор
k-ближайших соседей, *95*
Байеса, *464*; *607*
байесовский, *206*; *208*
линейный, *210*
параметрический, *212*

Классификация, *34*; *474*; *606*
двоичная, *147*
Кластеризация, *399*; *742*
Ключевой образ, *112*
Когерентная область, *612*
Кодирование, *864*
изображений, *544*
с минимальным искажением, *595*
Кодовое слово, *603*
Комбинаторная оптимизация, *710*
Компактное представление, *779*
Комплексное сопряжение, *68*
Конечный автомат, *926*; *937*
Конкурентное обучение, *23*; *79*; *101*; *573*
Конкуренция, *101*; *511*; *512*; *579*
Константа
Больцмана, *693*
времени поиска, *195*
Контекстная
информация, *35*
карта, *612*; *614*
Контекстные знания, *991*
Контекстный элемент, *923*
Контур с обратной связью, *176*
Контурная карта энергии, *862*
Концепт, *466*
Конъюнктивный синапс, *97*
Кооперация, *512*; *579*
Корреляционный синапс, *97*
Корреляция, *97*; *105*
взаимная, *542*
пространственная, *513*
Кофактор, *665*
Коэффициент
авторегрессии, *67*
Больцмана, *693*
доверия, *78*; *106*; *110*
корреляции, *610*
нормализации, *394*
обратной связи, *884*
отражения, *67*
передачи, *857*
разложения, *363*
усиления Калмана, *959*
Кратковременная память, *123*
Кривая
обучения, *192*
на проверочном подмножестве, *293*
Пеано, *599*
Критерий
минимакса, *191*
Неймана–Пирсона, *63*
правдоподобия, логарифмический, *209*
сходимости алгоритма обучения, *240*
Фишера, *273*
Кусочно-линейная функция, *46*

Л

Лагранжиан, *301*
Ландшафт энергии, *862*
Латеральное
взаимодействие, *580*
торможение, *101*
Лемма
об инвертировании матриц, *304*
Сауера, *153*
Линеаризация, *957*
Линейная
комбинация, *42*
адаптивная фильтрация, *173*
разделимость, *78*
регрессия, *490*
Линейное
отображение, *349*
преобразование, *514*
Линейный
адаптивный фильтр, *186*
ассоциатор, *890*
дискриминант Фишера, *272*; *274*
дифференциальный оператор, *356*
классификатор, *210*
поиск, *325*
сумматор, *46*; *121*
Линия передачи сигнала, *51*
Логарифмическая функция подобия, *493*
Логарифмический критерий правдоподобия, *209*
Логистическая

сеть доверия, *722*
функция, *46*; *682*
Ложный аттрактор, *871*
Локальная
обратная связь, *835*
цепочка, *39*
Локальное
рецепторное поле, *140*
усреднение, *331*
Локальный
градиент, *228*; *823*
минимум, *313*
признак, *287*

M

Максимальная неопределенность, *626*
Максимально правдоподобная оценка, *494*
Максимальное собственное значение, *530*
Максимизация
информативности, *246*
ожидания, *496*
Максимум взаимной информации, *82*; *622*
Марковская цепь, *696*
Массив антенных элементов, *121*
Масштабирование, *314*
Масштабирующий множитель, *594*
Математическое ожидание, *144*
Матрица
блокировки сигнала, *122*
вероятностей перехода, *697*
взвешенной
ковариации, *272*
нормы, *374*
влияния, *380*
внутриклассовой ковариации, *273*
Гессе, *180*; *276*
Грина, *365*
градиентов, *952*
данных, *560*
диагонально-доминантная, *889*
запоминания, *128*
измерений, *957*
интерполяции, *351*
ключей, *128*
ковариации, *209*
ошибки, *959*
корреляции, *77*; *185*; *515*; *530*
межклассовой ковариации, *273*
наблюдаемости, *934*
несингулярная, *351*
нижняя треугольная, *541*
обратная Гессиану, *302*
обучения, *77*
общей ковариации, *271*
ортогональная, *518*
памяти, *126*
перемешивания, *657*
положительно
определенная, *181*
полуопределенная, *888*
псевдообратная, *185*; *271*
симметричная, *516*; *866*; *888*
смешения, *655*
смешивания, *119*
соответствия, *611*
транспонированная, *61*
унитарная, *518*
управляемости, *932*
Якобиана, *182*
ядра, *564*
Машина
Больцмана, *81*; *104*; *691*; *713*
Гельмгольца, *728*
обучаемая, *110*; *141*
опорных векторов, *22*; *83*; *417*; *458*
Тьюринга, *939*
усиления, *471*
Межрегиональная цепочка, *39*
Мера
гладкости, *284*
емкости, *147*
избыточности, *648*
искажения, *589*; *743*
качества, *110*
новой информации, *958*
памяти, *187*
потерь, *142*
среднего объема информации, *625*

ссылки, *626*
эффективности, *296*; *334*
Метаустойчивое состояние, *586*
Метод
адаптивный, *562*
ассимптотически-оптимальный, *404*
Брента, *326*
Галеркина, *371*
Гаусса–Ньютона, *182*
градиентного спуска, *228*; *274*
квазиньютоновский, *318*; *327*
Ляпунова, *836*; *846*
ложных ближайших соседей, *901*
максимальной энтропии, *678*
минимизации структурного риска, *156*; *417*
множителей Лагранжа, *423*; *630*
моделирования отжига, *692*
Ньютона, *180*; *318*
наименьших квадратов, *22*; *373*; *543*
наискорейшего спуска, *235*
обучения с ранним остановом, *292*
пакетный, *560*
потенциальных функций, *83*
псевдообращения, *397*
радиальных базисных функций, *350*
регуляризации, *355*
сопряженных
градиентов, *319*; *322*
направлений, *319*; *321*
стохастической
аппроксимации, *195*; *312*
релаксации, *708*
удержания, *294*
уравнения ошибок, *955*
усиления, *465*
усреднения по ансамблю, *460*
Минимизация
квадратичной функции, *319*
риска
среднего, *207*
структурного, *156*; *417*
эмпирического, *144*; *146*
энергии, *854*
Минимум
глобальный, *240*
локальный, *240*; *313*
Многослойная нейронная сеть, *56*
Многослойный персептрон, *22*; *219*; *260*; *458*; *503*
архитектура, *274*
Многоступенчатое иерархическое векторное квантование, *608*
Множество
A-сопряженное, *319*
обучающее, *289*
тестовое, *289*
Множитель Лагранжа, *301*; *423*; *630*
Моделирование, *23*; *989*
отжига, *81*; *708*
Модель, *120*
BSB, *889*
NARX, *937*
антихеббовская, *97*
Вольтерра, *832*
в пространстве состояний, *837*; *923*
вероятностная, *80*
вероятностно-корректная, *158*
идентификации, *980*
иерархического смешения мнений экспертов, *484*
изинговская, *80*
информационно-теоретическая, *622*
Кохонена, *574*; *577*
Линскера, *513*
линейная, *524*
линейной регрессии, *490*
Мак-Каллока–Питца, *45*
HME, *492*
не-хеббовская, *97*
нейрона, *42*
Мак-Каллока–Питца, *196*
нелинейной
авторегрессии, *921*
регрессии, *446*
отображения признаков, *575*
прогнозирования, *170*
распознавания, *728*

регрессионная, *135*
селективного обучения Дарвина, *163*
симметричная, *559*
смешения мнений экспертов, *479*
состояния мозга, *884*
структурированная, *75*
Уилшоу–ван дер Мальсбурга, *577*
Хопфилда, *856*
хеббовская, *97*
эталонная, *979*
Модификация процесса конкуренции, *596*
Модульная сеть, *459*
нейронная, *477*
Мозг, *37*
Моментальная оценка, *108*
Мощность
вычислительная, *147*
канала, *648*
Мультиномиальная вероятность, *481*

Н

Наиболее правдоподобное значение, *494*
Наискорейший спуск, *669*
Направление
Ньютона, *327*
разбиения, *489*
Наращивание сети, *295*
Насыщение, *525*; *579*
Нат, *625*; *906*
Натуральный градиент, *669*
Начальное условие, *953*
Невзаимный четырехполюсник, *38*
Независимая переменная, *135*
Нейробиология, *36*; *575*
Нейродинамическое программирование, *761*; *793*
Нейроконтроллер, *117*
Нейроморфный контур, *37*
Нейрон, *31*; *38*; *39*; *42*
видимый, *105*
второго порядка, *926*
первого порядка, *926*
победивший, *573*
скрытый, *105*; *220*; *268*
стохастический, *713*
Нейрон-победитель, *101*; *579*
Нейронная сеть, *32*; *51*
аттракторная, *81*
линейная, *124*
многослойная, *56*
модульная, *477*
неполносвязная, *57*
однослойная, *56*
полносвязная, *57*
прямого распространения, *56*
рекуррентная, *36*; *57*
Нейронное изображение, *36*
Нейронный
микроконтур, *39*
фильтр, *815*
Нелинейная
динамическая система, *23*
функция активации, *220*
Нелинейное отображение, *349*
вход-выход, *281*
Нелинейность, *33*
Нелинейный адаптивный фильтр, *77*
Неопределенность, *626*
Неполная задача, *496*
Непрерывное обучение, *133*
Непрямое
обучение, *118*
управление, *979*
Неравенство
Гучи–Шварца, *202*
Йенсена, *507*; *735*
Несвязный расширенный фильтр Калмана, *963*
Нестационарная стратегия, *764*
Нечеткая система, *992*
Неявное представление, *799*
Нижний предел, *143*
Низкочастотный фильтр, *187*
Норма
взвешенная, *373*
Евклидова, *130*
Нормализация, *68*; *394*; *525*
входов, *248*

Нормализованная степень свободы, *583*
Нормализованный собственный вектор, *542*
Нуль-пространство, *366*

О

Область притяжения, *529*
Обобщающая способность, *278*
Обобщение, *33*; *60*
  логистической функции, *480*
Обобщенное дельта-правило, *236*
Обобщенный алгоритм
  Ллойда, *590*; *609*
  обучения Хебба, *538*
Обработка
  изображений, *23*; *331*
  информации, *73*
  параллельная, *73*
  сигналов, *173*; *989*
Образ
  активности, *909*
  запомненный, *112*
  ключевой, *112*
  линейно-разделимый, *211*
  нелинейно-разделимый, *433*
Обратная
  задача, *288*
  кинематика, *288*
  связь, *23*; *52*; *57*; *511*; *835*
Обратное распространение во времени, *981*
Обратный проход, *233*; *243*
Обусловленность, *371*
Обучаемая
  машина, *110*; *141*
  система, *762*
Обучаемый элемент, *72*
Обучающая выборка, *60*; *135*
Обучающее множество, *289*
Обучающие данные, *60*
Обучение, *32*; *60*; *72*; *89*; *868*
  анти-Хеббовское, *550*
  Больцмана, *104*; *163*
  без учителя, *23*; *108*; *110*; *509*
  гибридное, *399*
  избыточное, *279*
  конкурентное, *79*; *101*; *163*; *400*; *514*
  на временных разностях, *795*
  на лету, *133*
  на основе
    коррекции ошибок, *91*; *163*; *206*
    обратного распространения, *220*
    памяти, *93*; *163*
    самоорганизации, *110*; *399*; *509*; *510*
  непараметрическое, *34*
  непрерывное, *133*; *134*
  непрямое, *118*
  персептрона, *212*
  по подсказке, *252*
  полное, *300*
  прямое, *118*
  с подкреплением, *23*; *81*; *109*; *760*
    отложенным, *109*
  с ранним остановом, *292*
  с учителем, *22*; *33*; *107*; *399*
  самоорганизующееся, *509*
  статистическое, *34*
  Хебба, *163*; *549*
Общность, *131*
Обычное дифференциальное уравнение, *528*
Ограничение, *333*
  локальности, *306*
Одночлен, *344*; *439*
Ожидаемая ошибка, *135*
Ожидаемое искажение, *589*; *743*
Ожидаемый отклик, *176*
Оператор
  градиента, *177*
  единичной задержки, *53*; *93*; *189*; *526*; *801*
  замкнутого контура, *53*
  линейный дифференциальный , *356*
  псевдодифференциальный, *368*
  разомкнутого контура, *53*
  самосопряженный, *365*
  следа, *271*
  сопряженный, *360*
  статистического ожидания, *137*
  усреднения, *137*
Операция повторной оценки, *559*
Опорный вектор, *421*

Определение собственных значений, *518*
Оптимальная
  гиперплоскость, *419*
  степень повреждения, *301*
  хирургия мозга, *301*
Оптимизация
  безусловная, *177*
  численная, *317*
Оптическое распознавание символов, *469*
Опыт, *127*
Ортогональное преобразование подобия, *519*
Ортогональный набор, *131*
Ослабление, *99*
Отвод, *803*
Отказоустойчивость, *35*
Отклик, *129*
  желаемый, *205*
  импульсный, *802*
  ожидаемый, *176*
Относительная энтропия, *626*
Относительный градиент, *669*
Отношение
  правдоподобия, *208*; *257*
  сигнал/шум, *643*; *873*
Отображение
  вход-выход, *33*
  входа на выход, *854*
  линейное, *349*
  нелинейное, *349*; *562*
  признаков, *331*
Отрицательное биномиальное распределение, *348*
Отрицательный пример, *158*
Оценка
  достоверная, *900*
  несмещенная, *382*
  перекрестная, *382*
  плотности, *390*
  по методу наименьших квадратов, *543*
  состоятельная, *392*
  текущая, *312*
Очевидность ответа, *35*
Ошибка, *112*
  аппроксимации, *139*
  интегральная квадратичная, *284*
  классификации, *152*; *207*
  обобщения, *155*
  обучения, *152*; *155*
  ожидаемая, *135*
  оценивания, *139*; *285*; *388*
  резидуальная, *322*
  среднеквадратическая, *261*; *379*
  фильтрованной оценки, *959*

П

Пакетная версия метода градиентного спуска, *495*
Пакетное обучение, *966*
Пакетный
  алгоритм SOM, *593*
  режим, *238*; *246*
    обучения, *590*
Память, *122*; *800*
  автоассоциативная, *111*
  ассоциативная, *909*
  в виде матрицы корреляции, *128*
  гетероассоциативная, *111*
  долговременная, *123*; *800*
  кратковременная, *123*; *800*
  распределенная, *123*
  совершенно ассоциированная, *131*
Парадигма
  обучения, *90*
  связности, *306*
Параллельная
  модель идентификации, *976*
  распределенная обработка, *73*
Параллельное распределенное вычисление, *313*
Параметр
  доверия, *159*
  загрузки, *873*
  масштабирования, *403*
  наклона, *47*
  ошибки, *159*
  регуляризации, *297*; *357*
  свободный, *33*
  сглаживания, *413*

скорости обучения, *92*; *178*; *201*; *206*; *225*; *235*; *584*; *720*
Первичный сигнал подкрепления, *109*
Первый абсолютный момент, *284*
Передаточная функция, *49*; *802*
Перекрестная
оценка, *382*
проверка, *289*
многократная, *294*
обобщенная, *383*
Переменная
зависимая, *135*
независимая, *135*
фиктивная, *447*
Переменная-индикатор, *499*
Переобучение, *279*; *292*
Переучивание, *134*
Переход состояния, *926*
Персептрон, *78*; *196*
многослойный, *219*; *260*
Розенблатта, *172*; *196*
Пирамидальная клетка, *38*
Пластичность, *32*
синапсов, *512*
Плотность дискретизации, *286*
Победивший нейрон, *573*
Поверхность
ошибки, *107*
разделяющая, *344*
энергии, *880*
Повторное использование данных обучения, *470*
Подвыборка, *331*
Подмножество
для оценивания, *289*
проверочное, *289*
Подпространство главных компонентов, *559*
Поиск, *72*
линейный, *325*
максимального подобия, *587*
Покрытие Маркова, *739*
Поле, *164*
восприятия, *513*
рецептивное, *331*; *375*
рецепторное, *64*
чувствительности, *375*
Полезный сигнал, *536*
Полином Эрмита, *662*
Полное обучение, *294*
Положительный пример, *158*
Помеха, *67*
обработки, *634*
Понятие, *158*; *466*
целевое, *158*
Попытка Бернулли, *348*
Порог, *225*; *576*; *579*
критерия, *257*
Порождающая модель, *696*; *728*
Портрет состояний, *839*
Порядок
модели, *67*; *923*
памяти, *802*
системы, *928*
Последовательная обработка, *73*
Последовательно-параллельная модель идентификации, *976*
Последовательный
режим, *238*
Постоянная момента, *236*
Постпроцессор, *68*
Постулат
обучения, *76*
Хебба, *95*; *128*; *524*; *866*
Потенциал
активации, *44*
действия, *39*
Поток, *898*
динамической системы, *839*
Правило, *72*
адаптации с фиксированным приращением, *201*
байесовское, *255*
ближайшего соседа, *94*; *590*; *604*
большинства, *880*
Видроу–Хоффа, *91*
внешнего произведения, *128*
классификации, байесовское, *259*
конкурентного обучения, *103*

обучения
Больцмана, *105*; *720*
на основе коррекции ошибок, *206*
сигмоидальной сети доверия, *726*
Хебба, *549*; *721*
сохранения информации, *485*
умножения активности, *98*
цепочки, *227*; *681*
Предел
верхний, *143*
нижний, *143*
Чернова, *262*
Предикат, *314*
Представление, *71*
знаний, *58*; *72*
избыточное , *78*
мысленное, *74*
символьное, *73*
Преобразование
аффинное, *374*
Карунена–Лоева, *515*
линейное, *514*
Препроцессор, *68*; *569*
Приближенная функция стоимости перехода, *778*
Приближенный
алгоритм итерации по стратегиям, *780*
анализ данных, *688*
Признак
глобальный, *287*
инвариантный, *66*
локальный, *287*
Пример, *158*
маркированный, *59*
немаркированный, *59*
отрицательный, *60*; *158*
положительный, *60*; *158*
Принцип
Infomax, *163*; *642*
бритвы Оккама, *474*
детального баланса, *704*
максимального правдоподобия, *718*
максимальной энтропии, *629*
максимума
взаимной информации, *622*; *641*
энтропии, *623*
минимизации эмпирического риска, *146*
минимума
избыточности, *647*
свободной энергии, *695*
оптимальности Беллмана, *766*
ортогональности, *136*; *522*
полного количества информации, *163*
проекции, *80*
пространственного разнесения, *657*
разделяй и властвуй, *458*
регуляризации, *358*
самоорганизации, *511*
самоусиления, *579*
Присваивание коэффициентов доверия, *79*; *106*
Проба Бернулли, *698*
Проблема
избыточного подбора, *353*
обращения градиентов в нуль, *969*
Прогнозирование, *118*
временных рядов, *829*
Проекция, *515*
вектора, *520*
Произведение
векторов, *129*
внешнее, *127*
Проклятие размерности, *286*; *778*
Прореживание, *750*
Простая
модель, *503*
рекуррентная сеть, *923*
Пространственная решетка, *79*
Пространственно непрерывное входное пространство, *588*
Пространственно-кодированное распределение вероятности, *574*
Пространственное
различие, *652*
разнесение, *657*
Пространство, *928*
входное, *174*; *928*
выходное, *928*

гипотез, *159*
данных, *514*
наблюдений, *113*
нормированное, *356*
понятий, *158*
признаков, *268*; *344*; *433*; *514*
решений, *113*
Соболева, *387*
скрытое, *268*; *344*
состояний, *695*; *928*
функциональное, *356*
Прототип, *253*
Проход
обратный, *219*; *233*; *243*
прямой, *219*; *232*; *242*
Процедура
анализа, *520*
байесовская, *206*
исключения весов, *298*
обучения с сохранением информации, *965*
оценивания, оптимальная робастная, *445*
подбора, *350*
кривых, *325*
проверки гипотез, *206*
синтеза, *520*
снижения весов, *297*
Процесс
авторегрессионный, *67*
адаптации, *176*
взаимодействия, *622*
диссипативный, *898*
инновации, *958*
мыслительный, *74*
обучения, *123*
псевдостационарный, *133*
синаптической адаптации, *583*
фильтрации, *175*
Прямое обучение, *118*
Прямой
метод Ляпунова, *836*; *846*
проход, *232*; *242*
Псевдообратная матрица, *185*
Псевдотемпература, *105*; *694*

P

Равенство Чепмена–Колмогорова, *697*
Радиальная базисная функция, *82*
Разбиение, *148*
бинарное, *343*
входного пространства, *503*
Разделимость образов, *343*
Разложение
по собственным векторам, *519*
Холецкого, *960*
Размерность
входного пространства, *174*
сети, *125*
сокращение, *514*; *520*
Разностное уравнение, *527*
Разреженная архитектура, *991*
Разрешение памяти, *803*
Ранг
якобиана, неполный, *276*
Расписание отжига, *709*
Распознавание
образов, *113*; *989*
временное, *809*
структурное, *809*
цифр, *60*
Распределение
Гаусса, *395*
Гиббса, *691*; *693*
естественное, *895*
отрицательное биномиальное, *348*
среднего поля, *735*
Распределенная память, *123*
Рассогласование, *193*
Расстояние
алгебраическое, *421*
Евклидово, *61*
Махаланобиса, *62*
Рассуждения, *72*
Расходимость, *54*
Расширенная фильтрация Калмана, *981*
Расширенный полный набор данных, *496*
Рациональное многообразие, *344*
Регрессионная функция, *136*
Регрессия, *478*

временных рядов, *404*
ядра, *390*
Режим
асимптотический, *294*
интерактивный, *238*
неассимптотический, *294*
пакетный, *238*; *246*
последовательный, *238*; *246*
стохастический, *238*
Резидуальная ошибка, *322*
Резкость разбиения, *489*
Рекуррентная
сеть, *23*; *36*; *57*; *923*; *929*
второго порядка, *926*
Рекуррентное обучение в реальном времени, *949*; *981*
Рекуррентный многослойный персептрон, *925*; *967*
Ренормализованный алгоритм SOM, *621*
Репертуар, *164*
Репликатор, *307*
Рецептивное поле, *331*
Рецепторное поле, *64*; *513*
Решение
асимптотически устойчивое, *529*
Винеровское, *186*
корректное, *207*
минимальное по норме, *377*
некорректное, *207*
однозначное, *256*
Решетка, *573*
двумерная, *575*
Решетчатый фильтр, *67*
Риск
гарантированный, *155*
средний, *207*
структурный, *156*
эмпирический, *146*; *284*
Ряд
со скользящим средним, *832*
Тейлора, *300*

С

Самоорганизация, *509*; *511*; *576*
Самоорганизующаяся карта, *577*
Самоорганизующееся обучение, *760*
Самоусиление, *549*
Свертка, *331*; *817*
Свободная энергия, *694*
Свободное состояние, *105*
Свободные параметры, *33*
Свойство
Маркова, *696*
наилучшей аппроксимации, *371*
унитарной инвариантности, *455*
Связность, *220*
анатомическая, *909*
функциональная, *909*
Связь, *42*
возбуждающая, *102*; *244*
глобальная, *919*
латеральная, *550*
локальная, *919*
обратная, *550*; *835*
прямая, *549*
тормозящая, *244*; *550*
Сглаживание, *118*
Седловая точка, *251*; *423*
Семантическая карта, *614*
Семиинвариант, *663*
Сепаратриса, *849*
Сетчатка, *36*
Сеть
NWRE, *404*
RBF, *389*
ацикличная, *56*
внешне рекуррентная, *977*
внимания, *994*
возбуждения-торможения, *911*
доверия, *724*
классификации, *113*
модульная, *459*
на основе радиальных базисных функций, *22*; *95*; *342*
наблюдаемая, *931*
нейронов, *37*
обобщенная, *374*
однослойная, *56*

полносвязная, *222*
прямого распространения, *56*; *219*
размерность, *125*
регуляризации, *369*; *371*
рекуррентная, *52*; *919*
с задержкой по времени, *807*
свертки, *65*; *330*
связей, *33*
сетей, *908*
специализированная, *63*
управляемая, *931*
Хопфилда, *79*; *80*; *856*
шлюза, *479*
экспертов, *479*
Сжатие, *43*
данных, *309*; *577*
с потерей данных, *607*
Сигмоидальная
нелинейность, *234*
сеть доверия, *81*; *722*
функция, *46*
Сигнал, *49*
возбуждения, *109*
входной, *59*; *526*
выходной, *91*
линейно-разделимый, *172*
ошибки, *91*; *107*; *120*; *206*; *219*; *223*; *224*
подкрепления, *109*
полезный, *536*
постсинаптический, *93*
предсинаптический, *93*
функциональный, *223*
целевой, *176*
эталонный, *117*
Сигнум, *48*; *204*
Сила, *42*
Сильная модель обучения, *466*
Символьное представление, *73*
Символьный
код, *613*
язык, *71*
Симметричная синаптическая связь, *713*
Синапс, *38*; *39*; *42*
возбуждающий, *544*
конъюнктивный, *97*
корреляционный, *97*
тормозящий, *544*
Хебба, *96*
Синаптическая
адаптация, *579*; *583*
модификация, *76*
связь, *49*
Синаптический вес, *32*
Система
SISO, *931*
автономная, *533*
адаптивная, *76*
восприятия, *647*
гибридная, *75*
детерминированная, *893*
динамическая, *174*
инверсная, *116*
классификации, *990*
линейная, *417*
нечеткая, *992*
подавления боковых лепестков, *122*
прогнозирования, одношаговая, *812*
робастная, *445*
с обратной связью, *93*
положительной, *884*
с одной обратной связью, *52*
стохастическая, *189*
управления с обратной связью, *117*; *977*
хаотическая, *893*
Скалярное произведение, *61*
Скованное состояние, *105*
Скорость
обучения, *92*; *98*; *178*; *253*
сходимости, *193*
Скрученная карта, *620*
Скрытая модель Маркова, *809*
Скрытое пространство, *268*
Скрытый
нейрон, *56*; *220*; *268*
слой, *56*; *219*
элемент, *56*
Слабая модель обучения, *467*
Слагаемое

забывания, *583*
регуляризации, *356*
стандартной ошибки, *355*
След, *190*
матрицы, *190*
оператор, *271*
Слепое
разделение
входных сигналов, *596*
источников сигнала, *623*
сигнала, *119*
уравнивание, *686*
Сложная модель, *503*
Сложность
вычислительная, *162*; *310*
класса аппроксимирующих функций, *386*
обучающего множества, *160*
полиномиальная, *310*
Слой
входной, *219*; *509*
выходной, *219*; *509*
скрытый, *219*; *342*
Случайный вектор, *515*
Смешанная нестационарная стратегия, *789*
Смешанное состояние, *880*
Смешение мнений экспертов, *459*; *479*
Смещение, *139*
Снижение размерности, *600*
Собственная функция, *436*
Собственное значение, *190*; *436*; *518*; *523*; *530*; *543*
доминирующее, *521*
Собственный вектор, *523*; *542*
Совместная
апостериорная вероятность, *491*
фильтрация, *469*
энтропия, *633*
Совместное использование весов, *64*; *140*; *376*
Согласованность, *160*
Сокращение размерности, *514*; *520*
Сообщение, *624*
Сопряженный оператор, *360*
Состояние, *762*
нелинейного фильтра, *812*
прототипов, *864*
равновесия, *931*
равновесное, *842*
системы, *695*
динамической, *928*
смешанное, *880*
стационарное, *842*
устойчивое, *91*
Спектр, *70*
Спектральная теорема, *519*
Спектрограмма, *807*
Специализированная сеть, *63*
Сплайн, *288*
Способность
обобщающая, *278*
разделяющая, *349*
Справочная таблица, *609*; *787*
Среда, *141*
нестационарная, *133*; *562*
пространства состояний, *981*
стационарная, *132*; *185*
эргодическая, *560*
Среднее, *560*
взвешенное, *393*
время возврата, *699*
Среднеквадратическая
ошибка, *261*
сходимость, *190*
Средний риск, *207*
Средняя энергия, *694*
Стабилизатор, *356*
Стандартное искажение, *592*
Статистическая
механика, *23*; *691*
теория обучения, *22*
Статическая структура, *458*
Стационарная
среда, *132*
точка, *535*
Степень
выпуклости, *302*
повреждения, *301*
Стиль обработки, *73*

Стимул, *89*; *112*
Стохастическая
аппроксимация, *312*; *341*
матрица, *697*
релаксация, *708*
система, *189*
с обратной связью, *189*
Стохастический
градиент, *495*
нейрон, *713*
Стохастическое
моделирование, *692*
разностное уравнение, *527*
Стратегия, *764*
Строгая
интерполяция, *367*
состоятельность, *144*
Строка Пеано, *622*
Структура
временная, *70*
представления, *74*
Структурированная модель на основе связей, *75*
Структурная
задача, *106*
инвариантность, *66*
Структурное несоответствие, *677*
Структурный риск, *156*
Субоптимальная стратегия, *778*
Сумма
по всем состояниям, *693*
свертки, *65*; *801*; *820*
Сумматор, *42*
Сферическая симметрия, *394*
Схема
акцентирования, *247*
вложенная сигмоидальная, *310*
кодирования один-из-M, *272*
Сходимость, *54*
алгоритма GHA, *542*
в среднем, *189*
персептрона, *78*
по вероятности, *143*
среднеквадратическая, *190*

T

Текстура, *546*
Текущая оценка, *312*
Температура системы, *694*
Температурное равновесие, *714*
Теорема
вложения с задержкой, *899*
Дармуса, *688*
двойственности, *424*
Ковера, *343*; *346*
Коэна–Гроссберга, *882*
линейного отклика, *751*
Мерсера, *435*
Мичелли, *352*
о локальной
наблюдаемости, *935*
управляемости, *934*
о разделимости, *346*
образов, *343*
о сингулярной декомпозиции, *561*
о сходимости, *542*
персептрона, *173*
об асимптотической устойчивости, *529*
об обратной функции, *933*
об универсальной аппроксимации, *282*; *287*; *386*
об эргодичности, *701*
Ритца, *359*
спектральная, *519*
существования, *283*
сходимости персептрона, *78*; *203*
Теория
адаптивного резонанса, *79*
векторного квантования, *589*
вероятности, *718*
информации, *23*; *607*; *622*
коммуникаций, *577*
независимости, *190*
обучения, *140*
отбора групп нейронов, *163*
регрессии ядра, *343*
регуляризации, *22*; *122*
Тихонова, *297*; *355*; *904*
сложности, *911*

среднего поля, *692*; *730*
уровня искажения, *607*
Термальное равновесие, *105*
Терминальный узел, *486*
Тестовое множество, *289*
Тождество Грина, *360*
Топографическая карта, *39*; *573*
искусственная, *575*
Топологическая окрестность, *581*
Топологически упорядоченная вычислительная карта, *574*
Топологическое упорядочивание пространства признаков, *584*
Торможение, *38*
Тормозящий механизм дальнего действия, *576*
Точечный аттрактор, *848*
Точка
сингулярная, *842*
стационарная, *535*
Точность
наилучшей аппроксимации, *285*
эмпирического соответствия аппроксимации, *285*
Транзитное состояние, *698*

У

Узел, *49*
источника, *51*
Уменьшение избыточности, *647*
Универсальный аппроксиматор, *310*; *458*; *778*
Упорядоченность отображения, *575*
Управление, *72*; *989*
предприятием, *116*
Управляющее воздействие, *72*
Упрощение структуры сети, *296*
Уравнение
измерения, *928*; *957*
кабеля, *39*
оптимальности Беллмана, *770*
правдоподобия, *493*
процесса, *928*; *957*
Рикатти, *960*
среднего поля, *739*
Эйлера–Лагранжа, *361*; *363*
Уровень
активности, *511*
обобщения, *262*
объяснения, *73*
связности, *511*
Усеченный алгоритм обратного распространения, *947*
Усиление, *22*; *99*; *459*
за счет фильтрации, *467*
учителем, *955*
Ускорение спуска, *237*
Условие
выравнивания, *867*
Куна–Такера, *423*
Лившица, *841*
оптимальности, *423*
устойчивости, *867*
Условная энтропия, *632*
дифференциальная, *635*
Усреднение, *206*
локальное, *331*
по ансамблю, *22*; *459*; *460*
по времени, *98*
Установившееся значение вероятности, *699*
Устойчивое состояние, *91*; *867*
Устойчивость, *54*
глобальная, *847*
локальная, *843*; *847*
по Ляпунову, *836*
Учебный пример, *33*
Учитель, *141*

Ф

Фаза
восстановления, *112*
группировки, *325*
запоминания, *112*
обобщения, *350*
обучения, *350*
разделения, *325*
сохранения, *866*

Фактор
забывания, *527*
чувствительности, *227*
Факториальное распределение, *637*; *737*
Факториальный код, *647*
Феномен дивергенции, *960*
Физическое изображение, *36*
Фильтр, *118*
адаптивный, *173*; *175*
Винера, *186*
линейный, *813*
адаптивный, *133*
на основе наименьших квадратов, *184*
нелинейный адаптивный, *77*; *134*
низкочастотный, *187*
решетчатый, *67*
с конечной импульсной характеристикой, *815*
согласованный, *537*
Фильтрация, *118*; *467*
Калмана, *956*
пространственная, *120*
Фокусировка, *991*
Фонема, *806*
Формула
обратного распространения, *231*
Полака–Рибьера, *323*
синтеза, *520*
Флетчера–Ривза, *323*
Фрактальное измерение, *894*
Функционал
риска, *143*; *152*
эмпирического, *143*; *152*; *254*
Тихонова, *357*
Функциональный элемент, *72*
Функция
активации, *43*; *225*; *233*; *247*; *666*
нелинейная, *220*
антисимметричная, *247*
взвешивания, *393*
вычисления знака, *204*
Гаусса, *352*; *367*; *581*
изотропная, *397*
Грина, *365*
гиперболического тангенса, *235*; *247*
детерминированная, *135*
дискриминантная, *271*; *419*
единичного скачка, *45*
индикатора, *428*
коррекции, *528*
корреляции, *896*
Ляпунова, *533*; *846*; *858*
логарифмического правдоподобия на неполных данных, *497*
логистическая, *220*; *235*; *247*; *682*
многих переменных, *350*
мультиквадратичная, *352*
обратимая, *288*
обратная мультиквадратичная, *352*
обучения на подмножестве оценивания, *293*
окрестности, *582*
оценки регрессии, *393*
ошибки, *874*
передаточная, *802*
плотности вероятности, *627*
подобия, *493*
положительно-определенная, *370*
пороговая, *940*
потерь, *418*; *445*
правдоподобия, *718*
радиальная базисная, *341*; *350*
нормализованная , *394*
разбиения, *693*; *716*
регрессионная, *136*
роста, *148*
с односторонним насыщением, *940*
сигмоидальная, *220*
симметричная, *365*
собственная, *436*
стоимости, *91*; *109*; *226*; *821*
перехода, *765*
унимодальная, *325*
Хэвисайда, *45*; *896*
четная, *247*
четности, *314*
энергии, *105*; *858*

Х

Хаос, *83*
Характеристический код, *612*
Химический синапс, *38*
Хопфилд, *856*

Ц

Целевое значение, *248*
Целевой сигнал, *176*
Центр разложения, *363*
Центральная
  нервная система, *40*
  предельная теорема, *642*; *872*
Центробежный элемент, *37*
Цепное
  правило, *227*; *951*
    вычислений, *971*
Цепь Маркова, *692*; *696*

Ч

Частная производная, *274*
Частота эхо-сигнала, *70*
Частотная модуляция, *69*
Частотно-временной анализ, *993*
Численная оптимизация, *317*
Число
  обусловленности, *192*; *371*
  сглаживания, *290*
Чувствительность, *275*; *311*
  хаотического процесса, *897*

Ш

Шифратор, *521*
Шлюзовая сеть, *459*; *479*
Шум
  белый, *536*
  процесса, *965*
Шумоподавление, *829*

Э

Эквивариантная система, *668*
Эксперт, *458*; *479*
Экспертная система, *73*
Экспонента Ляпунова, *897*
Элемент
  единичной задержки, *58*
  памяти, *93*
  скрытый, *56*
  управления, *939*
  функциональный, *72*
Эмерджентность, *83*; *865*
Эмпирический риск, *146*; *284*
Эмпирическое
  знание, *135*
  правило Видроу, *281*
  среднее значение, *145*
Энергия
  максимальная, *311*
  ошибки, *91*; *224*
    среднеквадратической, *225*
  системы, *693*
Энтропия, *625*; *691*; *694*
  Шеннона, *744*
Эпоха, *226*; *238*; *261*
Эргодическая стационарная среда, *185*
Эргодичность, *699*
Эталонный сигнал, *117*; *979*
Эффект стабилизации, *237*
Эффективная ширина топологической
    окрестности, *581*
Эффективность классификации, *263*
Эффектор, *37*
Эхо-локация, *31*

Я

Ядро, *391*
  образующее, *802*
  положительно определенное, *436*
  скалярного произведения, *433*; *435*; *453*;
    562
Язык
  декларативный, *71*
  символьный, *71*
Якобиан, *182*; *276*; *842*
Ячейка
  Вороного, *603*
  фундаментальной памяти, *864*

Научно-популярное издание

Саймон Хайкин
Нейронные сети: полный курс
2-е издание

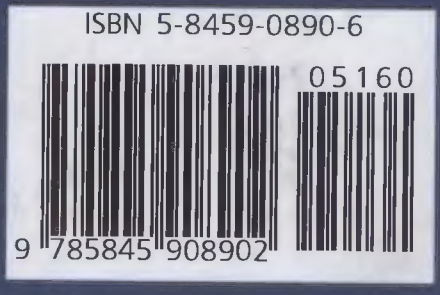
Литературный редактор	*И.А. Попова*
Верстка	*О.С. Назаренко*
Художественные редакторы	*В.Г. Павлютин, Т.А. Тараброва, О.А. Василенко*
Корректоры	*З.В. Александрова, Л.А. Гордиенко, О.В. Мишутина, Л.В. Чернокозинская*

Издательский дом "Вильямс".
101509, Москва, ул. Лесная, д. 43, стр. 1.

Подписано в печать 08.09.2005. Формат 70×100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 89,01. Уч.-изд. л. 57,11.
Тираж 2000 экз. Заказ № 6054.

Отпечатано с диапозитивов
в ФГУП "Печатный двор" им. А. М. Горького
Федерального агентства по печати
и массовым коммуникациям.
197110, Санкт-Петербург, Чкаловский пр., 15.

Издательский дом "Вильямс"
www.williamspublishing.com

PRENTICE HALL
Upper Saddle River, NJ 07458
http://www.prenhall.com

ISBN 5-8459-0890-6